F 1

```
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG
DDD         DDD  EEEEEEEEEEEE     BBBBBBBBBBBB     UUU         UUU  GGG
DDD         DDD  EEEEEEEEEEEE     BBBBBBBBBBBB     UUU         UUU  GGG
DDD         DDD  EEEEEEEEEEEE     BBBBBBBBBBBB     UUU         UUU  GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG   GGGGGGGGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG   GGGGGGGGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG   GGGGGGGGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG         GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG         GGG
DDD         DDD  EEE              BBB         BBB  UUU         UUU  GGG         GGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUU     GGGGGGGGG
```

```
                                                   C  2
**FILE**ID**DBGIFTHEN




DDDDDDDD    BBBBBBBB      GGGGGGGG    IIIIII    FFFFFFFFFF  TTTTTTTTTT  HH      HH  EEEEEEEEEE  NN      NN
DDDDDDDD    BBBBBBBB      GGGGGGGG    IIIIII    FFFFFFFFFF  TTTTTTTTTT  HH      HH  EEEEEEEEEE  NN      NN
DD      DD  BB      BB  GG              II      FF              TT      HH      HH  EE          NN      NN
DD      DD  BB      BB  GG              II      FF              TT      HH      HH  EE          NN      NN
DD      DD  BB      BB  GG              II      FF              TT      HH      HH  EE          NNNN    NN
DD      DD  BB      BB  GG              II      FF              TT      HH      HH  EE          NNNN    NN
DD      DD  BBBBBBBB    GG              II      FFFFFFFF        TT      HHHHHHHHHH  EEEEEEEE    NN  NN  NN
DD      DD  BBBBBBBB    GG              II      FFFFFFFF        TT      HHHHHHHHHH  EEEEEEEE    NN  NN  NN
DD      DD  BB      BB  GG  GGGGGG      II      FF              TT      HH      HH  EE          NN    NNNN
DD      DD  BB      BB  GG  GGGGGG      II      FF              TT      HH      HH  EE          NN    NNNN
DD      DD  BB      BB  GG      GG      II      FF              TT      HH      HH  EE          NN      NN
DD      DD  BB      BB  GG      GG      II      FF              TT      HH      HH  EE          NN      NN    ....
DDDDDDDD    BBBBBBBB      GGGGGG      IIIIII    FF              TT      HH      HH  EEEEEEEEEE  NN      NN    ....
DDDDDDDD    BBBBBBBB      GGGGGG      IIIIII    FF              TT      HH      HH  EEEEEEEEEE  NN      NN    ....
                                                                                                              ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
0 MODULE DBGIFTHEN (IDENT = 'V04-000') =
0
1 BEGIN
1
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1
1 !++
1 ! FACILITY:
1 !
1 !       DEBUG
1 !
1 ! ABSTRACT:
1 !
1 !       This module contains the parse and execution networks for the
1 !       DEBUG control structures: IF...THEN...ELSE, WHILE...DO,
1 !       FOR loops, and REPEAT...DO
1 !
1 ! ENVIRONMENT:
1 !
1 !       VAX/VMS
1 !
1 ! AUTHOR:
1 !
1 !       Richard Title
1 !
1 ! CREATION DATE:
1 !
1 !       1-10-82
1 !
1 ! VERSION:
1 !
1 !       V03.0-001
1 !
1 ! MODIFIED BY:
```

```
;      58          0058  1 !
;      59          0059  1 !
;      60          0060  1 ! REVISION HISTORY:
;      61          0061  1 !
;      62          0062  1 !--
;    64              0063  1 !
;    65              0064  1 ! TABLE OF CONTENTS:
;    66              0065  1 !
;    67              0066  1
;    68              0067  1 FORWARD ROUTINE
;    69              0068  1         DBG$NPARSE_IF,                          ! Parse network
;    70              0069  1         DBG$NEXECUTE_IF,                        ! Execution network
;    71              0070  1         DBG$NPARSE_WHILE,                       ! Parse network
;    72              0071  1         DBG$NEXECUTE_WHILE,                     ! Execution network
;    73              0072  1         DBG$NPARSE_FOR,                         ! Parse network for FOR
;    74              0073  1         DBG$NEXECUTE_FOR,                       ! Execution network for FOR
;    75              0074  1         DBG$NPARSE_REPEAT,                      ! Parse network
;    76              0075  1         DBG$NEXECUTE_REPEAT;                    ! Execution network
;    77              0076  1
;    78              0077  1 ! REQUIRE FILES:
;    79              0078  1 !
;    80              0079  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    81              0213  1 LIBRARY 'LIB$:DBGGEN.L32';
;    82              0214  1
;    83              0215  1 EXTERNAL
;    84              0216  1     dbg$gb_language: BYTE,              ! Current language setting
;    85              0217  1     dbg$gb_radix: VECTOR[3, BYTE],      ! Radix settings
;    86              0218  1     dbg$gb_take_cmd: BYTE,              ! Flag that controls command taking
;    87              0219  1     dbg$gl_cishead: REF cis$link;       ! Head of command input stream
;    88              0220  1
;    89              0221  1 EXTERNAL ROUTINE
;    90              0222  1     dbg$def_sym_add,                    ! Add a defined symbol
;    91              0223  1     dbg$get_memory,                     ! Allocate permanent memory
;    92              0224  1     dbg$get_tempmem,                    ! Allocates space
;    93              0225  1     dbg$ncis_add,                       ! Add a link to the command input stream
;    94              0226  1     dbg$nget_symid,                     ! Obtain a symid list
;    95              0227  1     dbg$nmake_arg_vect,                 ! Constructs error messages
;    96              0228  1     dbg$nmatch,                         ! Tries to match the next token
;    97              0229  1     dbg$nnext_word,                     ! Gets next word from input
;    98              0230  1     dbg$nparse_expression,              ! Language specific expression interpreter
;    99              0231  1     dbg$nread_name,                     ! Pick up a name
;   100              0232  1     dbg$nsave_break_buffer: NOVALUE,    ! Saves the action clause in a buffer
;   101              0233  1     dbg$nsyntax_error,                  ! Reports a syntax error
;   102              0234  1     dbg$ntype_conv,                     ! Language specific type converter
;   103              0235  1     dbg$rel_memory: NOVALUE,            ! Releases memory from DEBUG memory pool
;   104              0236  1     dbg$sta_lock_symid: NOVALUE;        ! Lock a symid list
;   105              0237  1
;   106            M 0238  1 MACRO report_error =
;   107            M 0239  1         BEGIN
;   108            M 0240  1         .message_vect =
;   109            M 0241  1             (IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;   110            M 0242  1              THEN
;   111            M 0243  1                 dbg$nmake_arg_vect(dbg$_needmore)
;   112            M 0244  1              ELSE
;   113            M 0245  1                  dbg$nsyntax_error (dbg$nnext_word (.input_desc)));
;   114            M 0246  1         RETURN sts$k_severe;
;   115              0247  1         END%;
```

```
GLOBAL ROUTINE dbg$nparse_if(input_desc, verb_node, message_vect) =
!
! Functional Description
!
!       ATN parse network for the IF verb.
!       This routine takes a verb node for the IF verb, and a string
!       descriptor for the remaining (unparsed) input.
!       A command execution tree is built. The form of the tree is:
!
!        -----------       -----------        -----------        -----------
!       | verb node |-->--| noun node |-->--| noun node |[-->--| noun node |]
!        -----------       -----------        -----------        -----------
!
!       The first noun node points to a value descriptor for the IF clause.
!       The second noun node points to a counted string with the THEN clause.
!       The third noun node, which may be absent, points to a counted string
!       with the ELSE clause.
!
! Formal Parameters
!
!       input_desc      - A longword containing the address of the
!                               command input descriptor.
!       verb_node       - A longword containing the address of the verb node.
!       message_vect    - The address of a longword to contain the address
!                               of a standard message argument vector.
!
! Implicit Inputs
!
!       none
!
! Implicit Outputs
!
!       On success, the command execution tree is constructed.
!       On failure, a message argument vector is constructed or obtained.
!
! Routine value
!
!       sts$k_success (1)       - Success. Command execution tree constructed.
!       sts$k_severe  (4)       - Failure. Error encountered. Message argument
!                                 constructed and returned.
!
! Side Effects
!
!       Permanent storage is allocated for the string holding the THEN
!       clause; this is released in DBG$NEXECUTE_IF after execution
!       of the THEN clause.
!

BEGIN

MAP
    input_desc  : REF dbg$stg_desc,
    verb_node   : REF dbg$verb_node;

BIND
    dbg$cs_cr           = UPLIT BYTE (1, dbg$k_car_return),
    dbg$cs_left_paren   = UPLIT BYTE (1, dbg$k_left_parenthesis),
    dbg$cs_else          = UPLIT BYTE (4, 'ELSE'),
    dbg$cs_then          = UPLIT BYTE (4, 'THEN');

LOCAL
    link,                                ! Temporary to links in the command
                                         !     execution tree.
    noun_node : REF dbg$noun_node,       ! A node in the command execution tree.
    radix,                               ! Holds the current radix setting.
    status;                              ! Holds return status from subroutine
                                         !     calls.



    ! Create and link a noun node
    !
    noun_node = dbg$get_tempmem(dbg$k_noun_node_size);
    verb_node[dbg$l_verb_object_ptr] = .noun_node;


    ! Determine the current radix.
    !
    radix  = .dbg$gb_radix[dbg$b_radix_input];


    ! Obtain a value descriptor for the condition. The first noun node
    ! points to this descriptor.
    !
    STATUS = DBG$NPARSE_EXPRESSION (.INPUT_DESC, .RADIX,
                     NOUN_NODE [DBG$L_NOUN_VALUE],
                     TOKEN$K_TERM_THEN, .MESSAGE_VECT);


    ! The return status should be "warning", meaning that an expression
    ! was parsed and further input reamins. If an expression was parsed
    ! but no input remains, then DBG$NPARSE_EXPRESSION will return success.
    ! In this context, it is an error since "IF condition" by itself
    ! is an error.
    !
    IF .status EQL sts$k_success THEN SIGNAL(DBG$_NEEDMORE);


    ! Severe status is also an error.
    !
    IF .status EQL sts$k_severe
    THEN
        RETURN sts$k_severe;


    ! Eat the THEN
    !
    IF NOT dbg$nmatch (.input_desc, dbg$cs_then, 1)
    THEN
        BEGIN
        .message_vect =
            (IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
              THEN
                 dbg$nmake_arg_vect (dbg$_needmore)
4               ELSE
3                   dbg$nsyntax_error (dbg$nnext_word (.input_desc)));
3
3           RETURN sts$k_severe;
2           END;
2
2
2   ! Allocate and link a noun node for the THEN clause.
2   !
2   link = noun_node [dbg$l_noun_link];
2   noun_node = dbg$get_tempmem(dbg$k_noun_node_size);
2   .link = .noun_node;
2
2   ! Eat the left parenthesis which we require be present.
2   !
2   IF NOT dbg$nmatch (.input_desc, dbg$cs_left_paren, 1)
2   THEN
3       BEGIN
3       .message_vect =
4           (IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
4            THEN
4               dbg$nmake_arg_vect (dbg$_needmore)
4            ELSE
5               BEGIN
5               SIGNAL(dbg$_needparen);
5               dbg$nsyntax_error (dbg$nnext_word (.input_desc))
3               END);
3       RETURN sts$k_severe;
2       END;
2
2   ! Put a pointer to the counted string representing the THEN
2   ! clause into the second noun node. (Note - the counted string
2   ! is constructed out of "permanent" memory which is released
2   ! in DBG$NEXECUTE_IF).
2   ! (The third argument indicates that this routine is not being
2   ! called during a SET BREAK DO (the behavior is slightly different
2   ! in that case.))
2   !
2   dbg$nsave_break_buffer (.input_desc, noun_node [dbg$l_noun_value]);
2
2   ! If we have reached the end of the line, return success (no else
2   ! clause is present).
2   !
2   IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
2      OR .input_desc [dsc$w_length] EQL 0
2   THEN
2       RETURN sts$k_success;
2
2   ! Look for ELSE clause.
2   !
2   IF NOT dbg$nmatch (.input_desc, dbg$cs_else, 1)
2   THEN
3       BEGIN
3       .message_vect = dbg$nsyntax_error (dbg$nnext_word (.input_desc));
3       RETURN sts$k_severe;
3       END;
2
```

```
;   288          0419  2        ! Allocate and link a noun node for the ELSE clause.
;   289          0420  2        !
;   290          0421  2        link = noun_node [dbg$l_noun_link];
;   291          0422  2        noun_node = dbg$get_tempmem(dbg$k_noun_node_size);
;   292          0423  2        .link = .noun_node;
;   293          0424  2
;   294          0425  2        ! Eat the left parenthesis which we require be present.
;   295          0426  2        !
;   296          0427  2        IF NOT dbg$nmatch (.input_desc, dbg$cs_left_paren, 1)
;   297          0428  2        THEN
;   298          0429  3            BEGIN
;   299          0430  3            .message_vect =
;   300          0431  4                (IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;   301          0432  4                 THEN
;   302          0433  4                    dbg$nmake_arg_vect (dbg$_needmore)
;   303          0434  4                 ELSE
;   304          0435  5                    BEGIN
;   305          0436  5                    SIGNAL(dbg$_needparen);
;   306          0437  5                    dbg$nsyntax_error (dbg$nnext_word (.input_desc))
;   307          0438  3                    END);
;   308          0439  3            RETURN sts$k_severe;
;   309          0440  2            END;
;   310          0441  2
;   311          0442  2        ! Put a pointer to the counted string representing the ELSE
;   312          0443  2        ! clause into the third noun node. (Note - the counted string
;   313          0444  2        ! is constructed out of "permanent" memory which is released
;   314          0445  2        ! in DBG$NEXECUTE_IF).
;   315          0446  2        !
;   316          0447  2        dbg$nsave_break_buffer (.input_desc, noun_node [dbg$l_noun_value]);
;   317          0448  2
;   318          0449  2        ! Return success.
;   319          0450  2        !
;   320          0451  2        RETURN sts$k_success;
;   321          0452  2
;   322          0453  1        END;


                                                                .TITLE  DBGIFTHEN
                                                                .IDENT  \V04-000\

                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                        0D  01  00000 P.AAA:    .BYTE   1, 13                                   ;
                                        28  01  00002 P.AAB:    .BYTE   1, 40                                   ;
                                            04  00004 P.AAC:    .BYTE   4                                       ;
                              45  53  4C  45  00005             .ASCII  \ELSE\                                  ;
                                            04  00009 P.AAD:    .BYTE   4                                       ;
                              4E  45  48  54  0000A             .ASCII  \THEN\                                  ;

                                                        DBG$CS_CR=              P.AAA
                                                        DBG$CS_LEFT_PAREN=      P.AAB
                                                        DBG$CS_ELSE=            P.AAC
                                                        DBG$CS_THEN=            P.AAD
                                                                .EXTRN  DBG$GB_LANGUAGE
                                                                .EXTRN  DBG$GB_RADIX, DBG$GB_TAKE_CMD
                                                                .EXTRN  DBG$GL_CISHEAD, DBG$DEF_SYM_ADD
                                                                .EXTRN  DBG$GET_MEMORY, DBG$GET_TEMPMEM
```

```
                                                                        .EXTRN   DBG$NCIS_ADD, DBG$NGET_SYMID
                                                                        .EXTRN   DBG$NMAKE_ARG_VECT
                                                                        .EXTRN   DBG$NMATCH, DBG$NNEXT_WORD
                                                                        .EXTRN   DBG$NPARSE_EXPRESSION
                                                                        .EXTRN   DBG$NREAD_NAME, DBG$NSAVE_BREAK_BUFFER
                                                                        .EXTRN   DBG$NSYNTAX_ERROR
                                                                        .EXTRN   DBG$NTYPE_CONV, DBG$REL_MEMORY
                                                                        .EXTRN   DBG$STA_LOCK_SYMID

                                                                        .PSECT   DBG$CODE,NOWRT,  SHR,  PIC,0

                                             07FC 00000                 .ENTRY   DBG$NPARSE_IF, Save R2,R3,R4,R5,R6,R7,R8,-   ; 0248
                                                                                 R9,R10                                       :
                                 5A 00000000G  00   9E 00002            MOVAB    DBG$NSAVE_BREAK_BUFFER, R10                  :
                                 59 00000000G  00   9E 00009            MOVAB    LIB$SIGNAL, R9                               :
                                 58 00000000G  00   9E 00010            MOVAB    DBG$GET_TEMPMEM, R8                          :
                                 57 00000000G  00   9E 00017            MOVAB    DBG$NMATCH, R7                               :
                                 56 00000000'  EF   9E 0001E            MOVAB    DBG$CS_CR, R6                                :
                                               04   DD 00025            PUSHL    #4                                           ; 0320
                                 68            01   FB 00027            CALLS    #1, DBG$GET_TEMPMEM                          :
                                 55            50   D0 0002A            MOVL     R0, NOUN_NODE                                :
                                 50      08    AC   D0 0002D            MOVL     VERB_NODE, R0                                ; 0321
                          08     A0            55   D0 00031            MOVL     NOUN_NODE, 8(R0)                             :
                                 50 00000000G  00   9A 00035            MOVZBL   DBG$GB_RADIX, RADIX                          ; 0326
                                 53      0C    AC   D0 0003C            MOVL     MESSAGE_VECT, R3                             ; 0334
                                               53   DD 00040            PUSHL    R3                                           :
                                               06   DD 00042            PUSHL    #6                                           ; 0333
                                               21   BB 00044            PUSHR    #^M<R0,R5>                                   :
                                 52      04    AC   D0 00046            MOVL     INPUT_DESC, R2                               ; 0332
                                               52   DD 0004A            PUSHL    R2                                           ; 0333
                   00000000G     00            05   FB 0004C            CALLS    #5, DBG$NPARSE_EXPRESSION                    :
                                 54            50   D0 00053            MOVL     R0, STATUS                                   :
                                 01            54   D1 00056            CMPL     STATUS, #1                                   ; 0343
                                               09   12 00059            BNEQ     1$                                           :
                                    000280D0   8F   DD 0005B            PUSHL    #164048                                      :
                                 69            01   FB 00061            CALLS    #1, LIB$SIGNAL                               :
                                 04            54   D1 00064  1$:       CMPL     STATUS, #4                                   ; 0348
                                               03   12 00067            BNEQ     2$                                           :
                                             00AE   31 00069            BRW      10$                                          :
                                               01   DD 0006C  2$:       PUSHL    #1                                           ; 0355
                                         09    A6   9F 0006E            PUSHAB   DBG$CS_THEN                                  :
                                               52   DD 00071            PUSHL    R2                                           :
                                 67            03   FB 00073            CALLS    #3, DBG$NMATCH                               :
                                 0E            50   E8 00076            BLBS     R0, 3$                                       :
                                               01   DD 00079            PUSHL    #1                                           ; 0359
                                       0044    8F   BB 0007B            PUSHR    #^M<R2,R6>                                   :
                                 67            03   FB 0007F            CALLS    #3, DBG$NMATCH                               :
                                 3D            50   E9 00082            BLBC     R0, 4$                                       :
                                               66   11 00085            BRB      6$                                           ; 0361
                                 54      08    A5   9E 00087  3$:       MOVAB    8(R5), LINK                                  ; 0371
                                               04   DD 0008B            PUSHL    #4                                           ; 0372
                                 68            01   FB 0008D            CALLS    #1, DBG$GET_TEMPMEM                          :
                                 55            50   D0 00090            MOVL     R0, NOUN_NODE                                :
                                 64            55   D0 00093            MOVL     NOUN_NODE, (LINK)                            ; 0373
                                               01   DD 00096            PUSHL    #1                                           ; 0377
                                         02    A6   9F 00098            PUSHAB   DBG$CS_LEFT_PAREN                            :
                                               52   DD 0009B            PUSHL    R2                                           :
```

```
                        67           03  FB 0009D        CALLS   #3, DBG$NMATCH                              ;
                        3E           50  E9 000A0        BLBC    R0, 5$                                      ;
                                     24  BB 000A3        PUSHR   #^M<R2,R5>                                  ; 0400
                        6A           02  FB 000A5        CALLS   #2, DBG$NSAVE_BREAK_BUFFER                  ;
                                     01  DD 000A8        PUSHL   #1                                          ; 0405
                             0044    8F  BB 000AA        PUSHR   #^M<R2,R6>                                  ;
                        67           03  FB 000AE        CALLS   #3, DBG$NMATCH                              ;
                        6F           50  E8 000B1        BLBS    R0, 12$                                     ;
                                     62  B5 000B4        TSTW    (R2)                                        ; 0406
                                     6B  13 000B6        BEQL    12$                                         ;
                                     01  DD 000B8        PUSHL   #1                                          ; 0412
                               04    A6  9F 000BA        PUSHAB  DBG$CS_ELSE                                 ;
                                     52  DD 000BD        PUSHL   R2                                          ;
                        67           03  FB 000BF        CALLS   #3, DBG$NMATCH                              ;
                        40           50  E9 000C2 4$:    BLBC    R0, 8$                                      ;
                        54     08    A5  9E 000C5        MOVAB   8(R5), LINK                                 ; 0421
                                     04  DD 000C9        PUSHL   #4                                          ; 0422
                        68           01  FB 000CB        CALLS   #1, DBG$GET_TEMPMEM                         ;
                        55           50  D0 000CE        MOVL    R0, NOUN_NODE                               ;
                        64           55  D0 000D1        MOVL    NOUN_NODE, (LINK)                           ; 0423
                                     01  DD 000D4        PUSHL   #1                                          ; 0427
                               02    A6  9F 000D6        PUSHAB  DBG$CS_LEFT_PAREN                           ;
                                     52  DD 000D9        PUSHL   R2                                          ;
                        67           03  FB 000DB        CALLS   #3, DBG$NMATCH                              ;
                        3D           50  E8 000DE        BLBS    R0, 11$                                     ;
                                     01  DD 000E1 5$:    PUSHL   #1                                          ; 0431
                             0044    8F  BB 000E3        PUSHR   #^M<R2,R6>                                  ;
                        67           03  FB 000E7        CALLS   #3, DBG$NMATCH                              ;
                        0F           50  E9 000EA        BLBC    R0, 7$                                      ;
                         000280D0    8F  DD 000ED 6$:    PUSHL   #164048                                     ; 0433
             00000000G  00           01  FB 000F3        CALLS   #1, DBG$NMAKE_ARG_VECT                      ;
                                     1B  11 000FA        BRB     9$                                          ;
                         00028743    8F  DD 000FC 7$:    PUSHL   #165699                                     ; 0436
                        69           01  FB 00102        CALLS   #1, LIB$SIGNAL                              ;
                                     52  DD 00105 8$:    PUSHL   R2                                          ; 0437
             00000000G  00           01  FB 00107        CALLS   #1, DBG$NNEXT_WORD                          ;
                                     50  DD 0010E        PUSHL   R0                                          ;
             00000000G  00           01  FB 00110        CALLS   #1, DBG$NSYNTAX_ERROR                       ;
                        63           50  D0 00117 9$:    MOVL    R0, (R3)                                    ; 0431
                        50           04  D0 0011A 10$:   MOVL    #4, R0                                      ; 0439
                                         04 0011D        RET                                                 ;
                                     24  BB 0011E 11$:   PUSHR   #^M<R2,R5>                                  ; 0447
                        6A           02  FB 00120        CALLS   #2, DBG$NSAVE_BREAK_BUFFER                  ;
                        50           01  D0 00123 12$:   MOVL    #1, R0                                      ; 0451
                                         04 00126        RET                                                 ; 0453

; Routine Size:  295 bytes,    Routine Base:  DBG$CODE + 0000
```

```
1 GLOBAL ROUTINE dbg$nexecute_if (verb_node,message_vect) =
1 !++
1 ! Functional Description
1 !
1 !       This routine performs the action associated with the IF
1 !       command.
1 !
1 ! Formal Parameters
1 !
1 !       verb_node        - A longword containing the address of the
1 !                          head (verb) node.
1 !       message_vect     - The address of a longword to contain the
1 !                          address of an error message vector
1 !
1 ! Implicit Inputs
1 !
1 !       The command tree contains a verb node and a linked list
1 !       of two or three noun nodes. (See the diagram in the header for
1 !       DBG$NPARSE_IF).
1 !
1 ! Routine Value
1 !
1 !       A completion code.
1 !
1 ! Completion Codes
1 !
1 !       sts$k_success (1)       - Success. Command executed
1 !       sts$k_severe (4)        - Failure. The command could not be
1 !                                 executed. An error message is constructed.
1 !
1 ! Side Effects
1 !
1 !       Storage allocated for the THEN clause is freed up.
1 !--
2     BEGIN
2
2     MAP
2         verb_node : REF dbg$verb_node;
2
2     LOCAL
2         condition_node: REF dbg$noun_node,      ! The noun node for the IF condition
2         condition_value,                        ! Should be TRUE or FALSE
2         else_node: REF dbg$noun_node,           ! The noun node for the ELSE clause.
2         else_string: REF VECTOR[,WORD],         ! Counted string with the ELSE clause
2         then_node:      REF dbg$noun_node,      ! The noun node for the THEN clause
2         then_string:    REF VECTOR[,WORD],      ! Counted string with the THEN clause
2         vax_desc:       dbg$stg_desc;           ! Target of the conversion from
2                                                 !    the value descriptor
2                                                 !    representing the condition.
2
2     ! Recover the two noun nodes.
2     !
2     condition_node = .verb_node [dbg$l_verb_object_ptr];
2     then_node = .condition_node [dbg$l_noun_link];
2     else_node = .then_node [dbg$l_noun_link];
2
2     ! Set up the vax descriptor for the condition.
```

```
2       ! *** For now, we just declare the descriptor to be longword integer,
2       ! since this causes the fewest problems in the type converter.
2       ! Eventually, if we get a Boolean type and all languages support
2       ! it then we will build a target descriptor of this type.
2       !
2       vax_desc [dsc$b_class] = dsc$k_class_s;
2       vax_desc [dsc$b_dtype] = dsc$k_dtype_l;
2       vax_desc [dsc$w_length] = 4;
2       vax_desc [dsc$a_pointer] = condition_value;
2       vax_desc [dsc$l_pos] = 0;
2
2       ! *** Special case for PASCAL. Level 3 PASCAL returns descriptors
2       ! of type Boolean (dsc$k_dtype_tf) for relational expressions.
2       !
2       IF .dbg$gb_language EQL dbg$k_pascal
2       THEN
3           BEGIN
3           vax_desc [dsc$b_dtype] = dsc$k_dtype_tf;
3           vax_desc [dsc$w_length] = 1;
2           END;
2
2       ! Initialize condition_value to 0
2       !
2       condition_value = 0;
2
2       ! Do the conversion from value descriptor to integer.
2       !
2       IF NOT dbg$ntype_conv (.condition_node [dbg$l_noun_value],
2                             dbg$k_default,
2                             dbg$k_vax_desc,
2                             vax_desc,
2                             .message_vect)
2       THEN
2           RETURN sts$k_severe;
2
2       ! Recover the string(s).
2       !
2       then_string = .then_node [dbg$l_noun_value];
2       IF .else_node NEQ 0
2       THEN
2           else_string = .else_node [dbg$l_noun_value]
2       ELSE
2           else_string = 0;
2
2       ! Process the THEN clause only if value of the condition is TRUE.
2       ! For now, just use the BLISS semantics which say that a value is
2       ! true iff the low bit is 1. We need to research which languages
2       ! have different semantics and come up with a language-dependent
2       ! method of doing this.
2       !
2       IF .condition_value
2       THEN
3           BEGIN
3
3           ! Add a new link to the command input stream.
3           !
3           IF NOT dbg$ncis_add (then_string[1], .then_string[0],
;   438         0568 3                          cis_if, 0, 0, 0, .message_vect)
;   439         0569 3          THEN
;   440         0570 3              RETURN sts$k_severe;
;   441         0571 3
;   442         0572 3          END
;   443         0573 3
;   444         0574 2      ELSE ! Process the ELSE clause
;   445         0575 2
;   446         0576 2          IF .else_string NEQ 0
;   447         0577 2          THEN
;   448         0578 3              BEGIN
;   449         0579 3
;   450         0580 3              ! Add a new link to the command input stream.
;   451         0581 3              !
;   452         0582 3              IF NOT dbg$ncis_add (else_string[1], .else_string[0],
;   453         0583 3                                   cis_if, 0, 0, 0, .message_vect)
;   454         0584 3              THEN
;   455         0585 3                  RETURN sts$k_severe;
;   456         0586 3
;   457         0587 2              END;
;   458         0588 2
;   459         0589 2      ! Return success.
;   460         0590 2      !
;   461         0591 2      RETURN sts$k_success;
;   462         0592 2
;   463         0593 1      END; ! dbg$nexecute_if



                                   000C 00000         .ENTRY  DBG$NEXECUTE_IF, Save R2,R3                       ; 0454
               5E            10   C2 00002         SUBL2   #16, SP                                           ;
               50       04   AC   D0 00005         MOVL    VERB_NODE, R0                                     ; 0506
               50       08   A0   D0 00009         MOVL    8(R0), CONDITION_NODE                             ;
               52       08   A0   D0 0000D         MOVL    8(CONDITION_NODE), THEN_NODE                      ; 0507
               53       08   A2   D0 00011         MOVL    8(THEN_NODE), ELSE_NODE                           ; 0508
     04        AE 01080004   8F   D0 00015         MOVL    #17301508, VAX_DESC                               ; 0518
     08        AE            6E   9E 0001D         MOVAB   CONDITION_VALUE, VAX_DESC+4                       ; 0519
                        0C   AE   D4 00021         CLRL    VAX_DESC+8                                        ; 0520
               06 00000000G  00   91 00024         CMPB    DBG$GB_LANGUAGE, #6                               ; 0525
                             08   12 0002B         BNEQ    1$                                                ;
     06        AE            28   90 0002D         MOVB    #40, VAX_DESC+2                                   ; 0528
     04        AE            01   B0 00031         MOVW    #1, VAX_DESC                                      ; 0529
                             6E   D4 00035 1$:     CLRL    CONDITION_VALUE                                   ; 0534
                        08   AC   DD 00037         PUSHL   MESSAGE_VECT                                      ; 0542
                        08   AE   9F 0003A         PUSHAB  VAX_DESC                                          ; 0538
               7E       82   8F   9A 0003D         MOVZBL  #130, -(SP)                                       ;
                             01   DD 00041         PUSHL   #1                                                ;
                             60   DD 00043         PUSHL   (CONDITION_NODE)                                  ;
  00000000G    00            05   FB 00045         CALLS   #5, DBG$NTYPE_CONV                                ;
               3B            50   E9 0004C         BLBC    R0, 6$                                            ;
               50            62   D0 0004F         MOVL    (THEN_NODE), THEN_STRING                          ; 0548
                             53   D5 00052         TSTL    ELSE_NODE                                         ; 0549
                             05   13 00054         BEQL    2$                                                ;
               52            63   D0 00056         MOVL    (ELSE_NODE), ELSE_STRING                          ; 0551
                             02   11 00059         BRB     3$                                                ;
```

```
                                          52  D4 0005B 2$:     CLRL    ELSE_STRING                              ; 0553
                        10                6E  E9 0005D 3$:     BLBC    CONDITION_VALUE, 4$                      ; 0561
                                  08      AC  DD 00060         PUSHL   MESSAGE_VECT                             ; 0568
                                          7E  7C 00063         CLRQ    -(SP)                                    ; 0567
                        7E                06  7D 00065         MOVQ    #6, -(SP)                                ;
                        7E                60  3C 00068         MOVZWL  (THEN_STRING), -(SP)                     ;
                                  02      A0  9F 0006B         PUSHAB  2(THEN_STRING)                           ;
                                          10  11 0006E         BRB     5$                                       ;
                                          1C  13 00070 4$:     BEQL    7$                                       ; 0576
                                  08      AC  DD 00072         PUSHL   MESSAGE_VECT                             ; 0583
                                          7E  7C 00075         CLRQ    -(SP)                                    ; 0582
                        7E                06  7D 00077         MOVQ    #6, -(SP)                                ;
                        7E                62  3C 0007A         MOVZWL  (ELSE_STRING), -(SP)                     ;
                                  02      A2  9F 0007D         PUSHAB  2(ELSE_STRING)                           ;
              00000000G 00                07  FB 00080 5$:     CALLS   #7, DBG$NCIS_ADD                         ;
                        04                50  E8 00087         BLBS    R0, 7$                                   ;
                        50                04  D0 0008A 6$:     MOVL    #4, R0                                   ; 0585
                                              04 0008D         RET                                              ;
                        50                01  D0 0008E 7$:     MOVL    #1, R0                                   ; 0591
                                              04 00091         RET                                              ; 0593

; Routine Size:  146 bytes,    Routine Base:  DBG$CODE + 0127
```

```
1 GLOBAL ROUTINE dbg$nparse_while(input_desc, verb_node, message_vect) =
1 !
1 ! Functional Description
1 !
1 !       ATN parse network for the WHILE verb.
1 !       This routine takes a verb node for the WHILE verb, and a string
1 !       descriptor for the remaining (unparsed) input.
1 !       A command execution tree is built. The form of the tree is:
1 !
1 !        -----------       -----------       -----------
1 !       | verb node |-->--| noun node |-->--| noun node |
1 !        -----------       -----------       -----------
1 !
1 !       The first noun node points to a value descriptor for the condition.
1 !       The second noun node points to a counted string with the DO clause.
1 !
1 ! Formal Parameters
1 !
1 !       input_desc      - A longword containing the address of the
1 !                               command input descriptor.
1 !       verb_node       - A longword containing the address of the verb node.
1 !       message_vect    - The address of a longword to contain the address
1 !                               of a standard message argument vector.
1 !
1 ! Implicit Inputs
1 !
1 !       none
1 !
1 ! Implicit Outputs
1 !
1 !       On success, the command execution tree is constructed.
1 !       On failure, a message argument vector is constructed or obtained.
1 !
1 ! Routine value
1 !
1 !       sts$k_success (1)       - Success. Command execution tree constructed.
1 !       sts$k_severe  (4)       - Failure. Error encountered. Message argument
1 !                                 constructed and returned.
1 !
1 ! Side Effects
1 !
1 !       Permanent storage is allocated for the string holding the DO clause;
1 !       this is released in DBG$NEXECUTE_WHILE after execution.
1 !
1
2   BEGIN
2
2   MAP
2       input_desc: REF dbg$stg_desc,
2       verb_node: REF dbg$verb_node;
2
2   BIND
2       dbg$cs_cr               = UPLIT BYTE (1, dbg$k_car_return),
2       dbg$cs_left_paren       = UPLIT BYTE (1, dbg$k_left_parenthesis),
2       dbg$cs_do               = UPLIT BYTE (4, 'DO');
2
2   LOCAL
```

```
2          link,                             ! Temporary to hold links in the command
2                                            !     execution tree.
2          noun_node : REF dbg$noun_node,    ! A node in the command execution tree.
2          radix,                            ! Holds the current radix setting.
2          status;                           ! Return status from subroutine calls.
2
2
2
2      ! Create and link a noun node
2      !
2      noun_node = dbg$get_tempmem(dbg$k_noun_node_size);
2      verb_node[dbg$l_verb_object_ptr] = .noun_node;
2
2      ! Determine the current radix.
2      !
2      radix = .dbg$gb_radix[dbg$b_radix_input];
2
2
2      ! Obtain a value descriptor for the condition. The first noun node
2      ! points to this value descriptor.
2      !
2      STATUS = DBG$NPARSE_EXPRESSION(.INPUT_DESC, .RADIX,
2                          NOUN_NODE [DBG$L_NOUN_VALUE],
2                          TOKEN$K_TERM_DO, .MESSAGE_VECT);
2
2
2      ! The return status should be "warning", meaning that an expression
2      ! was parsed and further input remains. If an expression was parsed
2      ! and no input remains, NPARSE_EXPRESSION will return "success".
2      ! In this context, it is an error since 'WHILE exp' by itself
2      ! is an error.
2      !
2      IF .status EQL sts$k_success
2      THEN
3          BEGIN
3          .message_vect = dbg$nmake_arg_vect (dbg$_needmore);
3          RETURN sts$k_severe;
3          END;
2
2      ! Severe status is also an error.
2      !
2      IF .status EQL sts$k_severe
2      THEN
2          RETURN sts$k_severe;
2
2      ! Eat the DO
2      !
2      IF NOT dbg$nmatch (.input_desc, dbg$cs_do, 1)
2      THEN
3          BEGIN
3          .message_vect =
4              (IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
4                  OR .input_desc [dsc$w_length] EQL 0
4               THEN
4                  dbg$nmake_arg_vect (dbg$_needmore)
4               ELSE
3                  dbg$nsyntax_error (dbg$nnext_word (.input_desc)));
```

```
;  579         0708  3             RETURN sts$k_severe;
;  580         0709  2             END;
;  581         0710  2
;  582         0711  2         ! Allocate and link a noun node for the DO clause.
;  583         0712  2         !
;  584         0713  2         link = noun_node [dbg$l_noun_link];
;  585         0714  2         noun_node = dbg$get_tempmem(dbg$k_noun_node_size);
;  586         0715  2         .link = .noun_node;
;  587         0716  2
;  588         0717  2         ! Eat the left parenthesis which we require be present.
;  589         0718  2         !
;  590         0719  2         IF NOT dbg$nmatch (.input_desc, dbg$cs_left_paren, 1)
;  591         0720  2         THEN
;  592         0721  3             BEGIN
;  593         0722  3             .message_vect =
;  594         0723  4                 (IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
;  595         0724  4                  THEN
;  596         0725  4                     dbg$nmake_arg_vect (dbg$_needmore)
;  597         0726  4                  ELSE
;  598         0727  5                     BEGIN
;  599         0728  5                     SIGNAL(dbg$_needparen);
;  600         0729  5                     dbg$nsyntax_error (dbg$nnext_word (.input_desc))
;  601         0730  3                     END);
;  602         0731  3             RETURN sts$k_severe;
;  603         0732  2             END;
;  604         0733  2
;  605         0734  2         ! Put a pointer to the counted string representing the DO
;  606         0735  2         ! clause into the second noun node. (Note - the counted string
;  607         0736  2         ! is constructed out of "permanent" memory which is released
;  608         0737  2         ! in DBG$NEXECUTE_IF).
;  609         0738  2         !
;  610         0739  2         dbg$nsave_break_buffer (.input_desc, noun_node [dbg$l_noun_value]);
;  611         0740  2
;  612         0741  2         ! Return success.
;  613         0742  2         !
;  614         0743  2         RETURN sts$k_success;
;  615         0744  2
;  616         0745  1         END;


                                                      .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                0D  01  0000E P.AAE:  .BYTE   1, 13                                    ;
                                28  01  00010 P.AAF:  .BYTE   1, 40                                    ;
                                    04  00012 P.AAG:  .BYTE   4                                        ;
                                4F  44  00013         .ASCII  \DO\                                     ;

                                              DBG$CS_CR=          P.AAE
                                              DBG$CS_LEFT_PAREN=  P.AAF
                                              DBG$CS_DO=          P.AAG

                                                      .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                  00FC  00000         .ENTRY  DBG$NPARSE_WHILE, Save R2,R3,R4,R5,R6,R7     ; 0594
                   57 00000000G  00  9E  00002         MOVAB   DBG$GET_TEMPMEM, R7                      ;
```

```
                56 00000000G  00  9E 00009        MOVAB   DBG$NMATCH, R6                      :
                55 00000000'  EF  9E 00010        MOVAB   DBG$CS_CR, R5                       :
                              04  DD 00017        PUSHL   #4                                  : 0661
                67            01  FB 00019        CALLS   #1, DBG$GET_TEMPMEM                 :
                54            50  D0 0001C        MOVL    R0, NOUN_NODE                       :
                50        08  AC  D0 0001F        MOVL    VERB_NODE, R0                       : 0662
          08    A0            54  D0 00023        MOVL    NOUN_NODE, 8(R0)                    :
                50 00000000G  00  9A 00027        MOVZBL  DBG$GB_RADIX, RADIX                 : 0666
                          0C  AC  DD 0002E        PUSHL   MESSAGE_VECT                        : 0674
                              05  DD 00031        PUSHL   #5                                  : 0673
                              11  BB 00033        PUSHR   #^M<R0,R4>                          :
                52        04  AC  D0 00035        MOVL    INPUT_DESC, R2                      : 0672
                              52  DD 00039        PUSHL   R2                                  : 0673
  00000000G     00            05  FB 0003B        CALLS   #5, DBG$NPARSE_EXPRESSION           :
                53            50  D0 00042        MOVL    R0, STATUS                          :
                01            53  D1 00045        CMPL    STATUS, #1                          : 0683
                              48  13 00048        BEQL    2$                                  :
                04            53  D1 0004A        CMPL    STATUS, #4                          : 0692
                              75  13 0004D        BEQL    6$                                  :
                              01  DD 0004F        PUSHL   #1                                  : 0698
                          04  A5  9F 00051        PUSHAB  DBG$CS_DO                           :
                              52  DD 00054        PUSHL   R2                                  :
                66            03  FB 00056        CALLS   #3, DBG$NMATCH                      :
                10            50  E8 00059        BLBS    R0, 1$                              :
                              01  DD 0005C        PUSHL   #1                                  : 0702
                              24  BB 0005E        PUSHR   #^M<R2,R5>                          :
                66            03  FB 00060        CALLS   #3, DBG$NMATCH                      :
                2C            50  E8 00063        BLBS    R0, 2$                              :
                              62  B5 00066        TSTW    (R2)                                : 0703
                              44  12 00068        BNEQ    4$                                  :
                              26  11 0006A        BRB     2$                                  : 0705
                53        08  A4  9E 0006C 1$:    MOVAB   8(R4), LINK                         : 0713
                              04  DD 00070        PUSHL   #4                                  : 0714
                67            01  FB 00072        CALLS   #1, DBG$GET_TEMPMEM                 :
                54            50  D0 00075        MOVL    R0, NOUN_NODE                       :
                63            54  D0 00078        MOVL    NOUN_NODE, (LINK)                   : 0715
                              01  DD 0007B        PUSHL   #1                                  : 0719
                          02  A5  9F 0007D        PUSHAB  DBG$CS_LEFT_PAREN                   :
                              52  DD 00080        PUSHL   R2                                  :
                66            03  FB 00082        CALLS   #3, DBG$NMATCH                      :
                40            50  E8 00085        BLBS    R0, 7$                              :
                              01  DD 00088        PUSHL   #1                                  : 0723
                              24  BB 0008A        PUSHR   #^M<R2,R5>                          :
                66            03  FB 0008C        CALLS   #3, DBG$NMATCH                      :
                0F            50  E9 0008F        BLBC    R0, 3$                              :
                   000280D0   8F  DD 00092 2$:    PUSHL   #164048                             : 0725
  00000000G     00            01  FB 00098        CALLS   #1, DBG$NMAKE_ARG_VECT              :
                              1F  11 0009F        BRB     5$                                  :
                   00028743   8F  DD 000A1 3$:    PUSHL   #165699                             : 0728
  00000000G     00            01  FB 000A7        CALLS   #1, LIB$SIGNAL                      :
                              52  DD 000AE 4$:    PUSHL   R2                                  : 0729
  00000000G     00            01  FB 000B0        CALLS   #1, DBG$NNEXT_WORD                  :
                              50  DD 000B7        PUSHL   R0                                  :
  00000000G     00            01  FB 000B9        CALLS   #1, DBG$NSYNTAX_ERROR               :
          0C    BC            50  D0 000C0 5$:    MOVL    R0, @MESSAGE_VECT                   : 0723
                50            04  D0 000C4 6$:    MOVL    #4, R0                              : 0731
                                  04 000C7        RET                                         :
```

```
                                                                              H  3
DBGIFTHEN                                                                16-Sep-1984 01:18:37    VAX-11 Bliss-32 V4.0-742            Page  18
V04-000                                                                  14-Sep-1984 12:16:59    [DEBUG.SRC]DBGIFTHEN.B32;1                (5)

                                                     14  BB 000C8 7$:      PUSHR   #^M<R2,R4>                                        ; 0739
                              00000000G  00          02  FB 000CA          CALLS   #2, DBG$NSAVE_BREAK_BUFFER                        ;
                                         50          01  D0 000D1          MOVL    #1, R0                                            ; 0743
                                                         04 000D4          RET                                                       ; 0745

; Routine Size:  213 bytes,    Routine Base:  DBG$CODE + 01B9
```

```
1 GLOBAL ROUTINE dbg$nexecute_while (verb_node,message_vect) =
1 !++
1 ! Functional Description
1 !
1 !       This routine performs the action associated with the WHILE
1 !       command.
1 !
1 ! Formal Parameters
1 !
1 !       verb_node       - A longword containing the address of the
1 !                         head (verb) node.
1 !       message_vect    - The address of a longword to contain the
1 !                         address of an error message vector
1 !
1 ! Implicit Inputs
1 !
1 !       The command tree contains a verb node and a linked list
1 !       of two noun nodes. (See the diagram in the header for
1 !       DBG$NPARSE_WHILE).
1 !
1 ! Routine Value
1 !
1 !       A completion code.
1 !
1 ! Completion Codes
1 !
1 !       sts$k_success (1)       - Success. Command executed
1 !       sts$k_severe (4)        - Failure. The command could not be
1 !                                 executed. An error message is constructed.
1 !
1 ! Side Effects
1 !
1 !       None
1 !--
2    BEGIN
2
2    MAP
2       verb_node : REF dbg$verb_node;
2
2    LOCAL
2       condition_node: REF dbg$noun_node,      ! The noun node for the IF condition
2       condition_value,                        ! Should be TRUE or FALSE
2       do_node:        REF dbg$noun_node,      ! The noun node for the THEN clause
2       do_string: REF VECTOR[,WORD],           ! Counted string for the do clause
2       vax_desc:  dbg$stg_desc;                ! Target of the conversion from
2                                               !       the value descriptor.
2
2
2    ! Recover the two noun nodes.
2    !
2    condition_node = .verb_node [dbg$l_verb_object_ptr];
2    do_node = .condition_node [dbg$l_noun_link];
2
2    ! Set up the vax descriptor for the condition.
2    !
2    vax_desc [dsc$b_class] = dsc$k_class_s;
2    vax_desc [dsc$b_dtype] = dsc$k_dtype_l;
```

```
;  675          0803  2       vax_desc [dsc$w_length] = 4;
;  676          0804  2       vax_desc [dsc$a_pointer] = condition_value;
;  677          0805  2       vax_desc [dsc$l_pos] = 0;
;  678          0806  2
;  679          0807  2       ! Special case for level 3 PASCAL. PASCAL returns descriptors
;  680          0808  2       ! of type boolean (dsc$k_dtype_tf) for relational expressions.
;  681          0809  2       !
;  682          0810  2       IF .dbg$gb_language EQL dbg$k_pascal
;  683          0811  2       THEN
;  684          0812  3           BEGIN
;  685          0813  3           vax_desc [dsc$b_dtype] = dsc$k_dtype_tf;
;  686          0814  3           vax_desc [dsc$w_length] = 1;
;  687          0815  2           END;
;  688          0816  2
;  689          0817  2       ! Initialize condition_value to zero.
;  690          0818  2       !
;  691          0819  2       condition_value = 0;
;  692          0820  2
;  693          0821  2       ! Do the conversion from value descriptor to integer.
;  694          0822  2       !
;  695          0823  2       IF NOT dbg$ntype_conv (.condition_node [dbg$l_noun_value],
;  696          0824  2                              dbg$k_default,
;  697          0825  2                              dbg$k_vax_desc,
;  698          0826  2                              vax_desc,
;  699          0827  2                              .message_vect)
;  700          0828  2       THEN
;  701          0829  2           RETURN sts$k_severe;
;  702          0830  2
;  703          0831  2       ! Continue only of condition is true. For now, just use BLISS semantics.
;  704          0832  2       !
;  705          0833  2       IF .condition_value
;  706          0834  2       THEN
;  707          0835  3           BEGIN
;  708          0836  3
;  709          0837  3           ! Recover the do string.
;  710          0838  3           !
;  711          0839  3           do_string = .do_node [dbg$l_noun_value];
;  712          0840  3
;  713          0841  3           ! Add a link to the command input stream
;  714          0842  3           !
;  715          0843  3           IF NOT dbg$ncis_add (do_string[1], .do_string[0], cis_while,
;  716          0844  3                   0, TRUE, 0, .message_vect)
;  717          0845  3           THEN
;  718          0846  3               RETURN sts$k_severe;
;  719          0847  3
;  720          0848  3           END
;  721          0849  3
;  722          0850  2       ELSE
;  723          0851  2           ! Add a cis for null action
;  724          0852  2           !
;  725          0853  3           BEGIN
;  726          0854  3           LOCAL
;  727          0855  3               dummy: REF VECTOR[,WORD];
;  728          0856  3           dummy = dbg$get_memory (1);
;  729          0857  3           IF NOT dbg$ncis_add (dummy[1], 0, cis_while, 0, FALSE, 0, .message_vect)
;  730          0858  3           THEN
;  731          0859  3               RETURN sts$k_severe;
;  732         0860  2          END;
;  733         0861  2
;  734         0862  2      ! Return success.
;  735         0863  2      !
;  736         0864  2      RETURN sts$k_success;
;  737         0865  2
;  738         0866  1      END; ! dbg$nexecute_while



                                          0004 00000          .ENTRY  DBG$NEXECUTE_WHILE, Save R2                    ; 0746
                      5E            10   C2 00002          SUBL2   #16, SP                                        ;
                      50      04    AC   D0 00005          MOVL    VERB_NODE, R0                                  ; 0796
                      50      08    A0   D0 00009          MOVL    8(R0), CONDITION_NODE                          ;
                      52      08    A0   D0 0000D          MOVL    8(CONDITION_NODE), DO_NODE                     ; 0797
                04    AE 01080004    8F   D0 00011          MOVL    #17301508, VAX_DESC                            ; 0803
                08    AE            6E   9E 00019          MOVAB   CONDITION_VALUE, VAX_DESC+4                    ; 0804
                              0C    AE   D4 0001D          CLRL    VAX_DESC+8                                     ; 0805
                      06 00000000G   00   91 00020          CMPB    DBG$GB_LANGUAGE, #6                            ; 0810
                                    08   12 00027          BNEQ    1$                                             ;
                06    AE            28   90 00029          MOVB    #40, VAX_DESC+2                                ; 0813
                04    AE            01   B0 0002D          MOVW    #1, VAX_DESC                                   ; 0814
                                    6E   D4 00031  1$:     CLRL    CONDITION_VALUE                                ; 0819
                              08    AC   DD 00033          PUSHL   MESSAGE_VECT                                   ; 0827
                              08    AE   9F 00036          PUSHAB  VAX_DESC                                       ; 0823
                      7E      82    8F   9A 00039          MOVZBL  #130, -(SP)                                    ;
                                    01   DD 0003D          PUSHL   #1                                             ;
                                    60   DD 0003F          PUSHL   (CONDITION_NODE)                               ;
          00000000G   00            05   FB 00041          CALLS   #5, DBG$NTYPE_CONV                             ;
                      34            50   E9 00048          BLBC    R0, 4$                                         ;
                      11            6E   E9 0004B          BLBC    CONDITION_VALUE, 2$                            ; 0833
                      50            62   D0 0004E          MOVL    (DO_NODE), DO_STRING                           ; 0839
                              08    AC   DD 00051          PUSHL   MESSAGE_VECT                                   ; 0844
                      7E            01   7D 00054          MOVQ    #1, -(SP)                                      ; 0843
                      7E            05   7D 00057          MOVQ    #5, -(SP)                                      ;
                      7E            60   3C 0005A          MOVZWL  (DO_STRING), -(SP)                             ;
                                    13   11 0005D          BRB     3$                                             ;
                                    01   DD 0005F  2$:     PUSHL   #1                                             ; 0856
          00000000G   00            01   FB 00061          CALLS   #1, DBG$GET_MEMORY                             ;
                              08    AC   DD 00068          PUSHL   MESSAGE_VECT                                   ; 0857
                                    7E   7C 0006B          CLRQ    -(SP)                                          ;
                      7E            05   7D 0006D          MOVQ    #5, -(SP)                                      ;
                                    7E   D4 00070          CLRL    -(SP)                                          ;
                              02    A0   9F 00072  3$:     PUSHAB  2(DUMMY)                                       ;
          00000000G   00            07   FB 00075          CALLS   #7, DBG$NCIS_ADD                               ;
                      04            50   E8 0007C          BLBS    R0, 5$                                         ;
                      50            04   D0 0007F  4$:     MOVL    #4, R0                                         ; 0859
                                    04 00082          RET                                                    ;
                      50            01   D0 00083  5$:     MOVL    #1, R0                                         ; 0864
                                    04 00086          RET                                                    ; 0866

; Routine Size:  135 bytes,    Routine Base:  DBG$CODE + 028E
```

```
1 GLOBAL ROUTINE dbg$nparse_for (input_desc, verb_node, message_vect) =
1 !
1 ! Functional Description
1 !
1 !       ATN parse network for the FOR verb.
1 !       This routine takes a verb node for the FOR verb, and a string
1 !       descriptor for the remaining (unparsed) input.
1 !       A command execution tree is built. The form of the tree is:
1 !
1 !        -----------        -----------        -----------         -----------
1 !       | verb node |-->--| noun node |-->--| noun node | -->-- | noun node |
1 !        -----------        -----------        -----------         -----------
1 !
1 !       The first noun node contains a counted string with the name of the
1 !       loop variable.
1 !       The second noun node contains value descriptors with the lower and
1 !       upper bounds, and loop increment
1 !       The third noun node contains a counted string with the command list.
1 !
1 ! Formal Parameters
1 !
1 !       input_desc      - A longword containing the address of the
1 !                               command input descriptor.
1 !       verb_node       - A longword containing the address of the verb node.
1 !       message_vect    - The address of a longword to contain the address
1 !                               of a standard message argument vector.
1 !
1 ! Implicit Inputs
1 !
1 !       none
1 !
1 ! Implicit Outputs
1 !
1 !       On success, the command execution tree is constructed.
1 !       On failure, a message argument vector is constructed or obtained.
1 !
1 ! Routine value
1 !
1 !       sts$k_success (1)       - Success. Command execution tree constructed.
1 !       sts$k_severe  (4)       - Failure. Error encountered. Message argument
1 !                                 constructed and returned.
1 !
1 ! Side Effects
1 !
1 !       Permanent storage is allocated for the string holding the action
1 !       clause and for the string holding the loop variable name.
1 !       This is released in DBG$NCIS_REMOVE after execution
1 !       of the action clause.
1 !
1
2   BEGIN
2
2   MAP
2       input_desc  : REF dbg$stg_desc,
2       verb_node       : REF dbg$verb_node;
2
2   BIND
```

```
2              dbg$cs_comma            = UPLIT BYTE (1, dbg$k_comma),
2              dbg$cs_cr               = UPLIT BYTE (1, dbg$k_car_return),
2              dbg$cs_equal            = UPLIT BYTE (1, dbg$k_equal),
2              dbg$cs_left_paren       = UPLIT BYTE (1, dbg$k_left_parenthesis),
2              dbg$cs_by               = UPLIT BYTE (2, 'BY'),
2              dbg$cs_do               = UPLIT BYTE (2, 'DO'),
2              dbg$cs_to               = UPLIT BYTE (2, 'TO');
2
2          LOCAL
2              link,                           ! Temporary to hold links in the command
2                                              !     execution tree.
2              noun_node : REF dbg$noun_node,  ! A node in the command execution tree.
2              radix,                          ! Holds the current radix setting.
2              status;                         ! Holds return status from subroutine
2                                              !     calls.
2
2
2          ! Create and link a noun node
2          !
2          noun_node = dbg$get_tempmem (dbg$k_noun_node_size);
2          verb_node[dbg$l_verb_object_ptr] = .noun_node;
2
2          ! Pick up the name of the loop counter.
2          ! Note that dbg$nread_name allocates permanent storage for the name.
2          ! This must be released in DBG$NCIS_REMOVE when the command buffer is
2          ! no longer needed.
2          !
2          IF NOT dbg$nread_name (.input_desc,
2                                 noun_node [dbg$l_noun_value],
2                                 .message_vect)
2          THEN
2              RETURN sts$k_severe;
2
2          ! Eat the =
2          !
2          IF NOT dbg$nmatch (.input_desc, dbg$cs_equal, 1)
2          THEN
2              report_error;
2
2          ! Create and link a noun node
2          !
2          link = noun_node [dbg$l_noun_link];
2          noun_node = dbg$get_tempmem (dbg$k_noun_node_size);
2          .link = .noun_node;
2
2          ! Determine the current radix.
2          !
2          radix  = .dbg$gb_radix[dbg$b_radix_input];
2
2          ! Obtain a value descriptor for the lower bound. The noun_value field
2          ! points to this descriptor.
2          !
2          STATUS = DBG$NPARSE_EXPRESSION(.INPUT_DESC, .RADIX,
2                                         NOUN_NODE [DBG$L_NOUN_VALUE],
2                                         TOKEN$K_TERM_TO, .MESSAGE_VECT);
2
2          ! The return status should be "warning", meaning that an expression
```

```
2      ! was parsed and further input reamins. If an expression was parsed
2      ! but no input remains, then DBG$NPARSE_EXPRESSION will return success.
2      ! In this context, it is an error since "REPEAT count" by itself
2      ! is an error.
2      !
2      IF .status EQL sts$k_success
2      THEN
3          BEGIN
3          .message_vect = dbg$nmake_arg_vect (dbg$_needmore);
3          RETURN sts$k_severe;
2          END;
2
2      ! Severe status is also an error.
2      !
2      IF .status EQL sts$k_severe
2      THEN
2          RETURN sts$k_severe;
2
2      ! Eat the "TO".
2      !
2      IF NOT dbg$nmatch (.input_desc, dbg$cs_to, 2)
2      THEN
2          report_error;
2
2      ! Obtain a value descriptor for the upper bound. The noun_value2 field
2      ! points to this descriptor.
2      !
2      STATUS = DBG$NPARSE_EXPRESSION(.INPUT_DESC, .RADIX,
2                                     NOUN_NODE [DBG$L_NOUN_VALUE2],
2                                     TOKEN$K_TERM_BY, .MESSAGE_VECT);
2
2
2      ! The return status should be "warning", meaning that an expression
2      ! was parsed and further input reamins. If an expression was parsed
2      ! but no input remains, then DBG$NPARSE_EXPRESSION will return success.
2      ! In this context, it is an error since "REPEAT count" by itself
2      ! is an error.
2      !
2      IF .status EQL sts$k_success
2      THEN
3          BEGIN
3          .message_vect = dbg$nmake_arg_vect (dbg$_needmore);
3          RETURN sts$k_severe;
2          END;
2
2      ! Severe status is also an error.
2      !
2      IF .status EQL sts$k_severe
2      THEN
2          RETURN sts$k_severe;
2
2      ! Check for BY clause.
2      !
2      IF dbg$nmatch (.input_desc, dbg$cs_by, 2)
2      THEN
3          BEGIN
3
```

```
	! Obtain a value descriptor for the increment.
	!
	STATUS = DBG$NPARSE_EXPRESSION(.INPUT_DESC, .RADIX,
				NOUN_NODE [DBG$L_ADJECTIVE_PTR],
				TOKEN$K_TERM_DO, .MESSAGE_VECT);

	! The return status should be "warning", meaning that an expression
	! was parsed and further input reamins. If an expression was parsed
	! but no input remains, then DBG$NPARSE_EXPRESSION will return success.
	! In this context, it is an error since "FOR I=1 TO N BY" by itself
	! is an error.
	!
	IF .status EQL sts$k_success
	THEN
	    BEGIN
	    .message_vect = dbg$nmake_arg_vect (dbg$_needmore);
	    RETURN sts$k_severe;
	    END;

	! Severe status is also an error.
	!
	IF .status EQL sts$k_severe
	THEN
	    RETURN sts$k_severe;

	END
    ELSE
	noun_node [dbg$l_adjective_ptr] = 0;

    ! Eat the "DO".
    !
    IF NOT dbg$nmatch (.input_desc, dbg$cs_do, 2)
    THEN
	report_error;

    ! Allocate and link a noun node for the action clause.
    !
    link = noun_node [dbg$l_noun_link];
    noun_node = dbg$get_tempmem (dbg$k_noun_node_size);
    .link = .noun_node;

    ! Eat the left parenthesis which we require be present.
    !
    IF NOT dbg$nmatch (.input_desc, dbg$cs_left_paren, 1)
    THEN
	report_error;

    ! Put a pointer to the counted string representing the action
    ! clause into the second noun node. (Note - the counted string
    ! is constructed out of "permanent" memory which is released
    ! in DBG$NCIS_REMOVE).
    ! The third argument indicates that save_break_buffer is not being
    ! called during parsing of a SET BREAK DO (The routine behaves
    ! slightly differently in that case)
    !
    dbg$nsave_break_buffer (.input_desc, noun_node [dbg$l_noun_value]);
```

```
;  968          1095  2
;  969          1096  2      ! Return success.
;  970          1097  2      !
;  971          1098  2      RETURN sts$k_success;
;  972          1099  2
;  973          1100  1      END;


                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                             2C  01  00015 P.AAH:   .BYTE   1, 44                                     ;
                             0D  01  00017 P.AAI:   .BYTE   1, 13                                     ;
                             3D  01  00019 P.AAJ:   .BYTE   1, 61                                     ;
                             28  01  0001B P.AAK:   .BYTE   1, 40                                     ;
                                 02  0001D P.AAL:   .BYTE   2                                         ;
                             59  42  0001E          .ASCII  \BY\                                      ;
                                 02  00020 P.AAM:   .BYTE   2                                         ;
                             4F  44  00021          .ASCII  \DO\                                      ;
                                 02  00023 P.AAN:   .BYTE   2                                         ;
                             4F  54  00024          .ASCII  \TO\                                      ;

                                           DBG$CS_COMMA=        P.AAH
                                           DBG$CS_CR=           P.AAI
                                           DBG$CS_EQUAL=        P.AAJ
                                           DBG$CS_LEFT_PAREN=   P.AAK
                                           DBG$CS_BY=           P.AAL
                                           DBG$CS_DO=           P.AAM
                                           DBG$CS_TO=           P.AAN


                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                  0FFC 00000            .ENTRY  DBG$NPARSE_FOR, Save R2,R3,R4,R5,R6,R7,R8,-  ; 0867
                                                                R9,R10,R11                                   ;
                5B 00000000G  00   9E 00002             MOVAB   DBG$GET_TEMPMEM, R11                         ;
                5A 00000000G  00   9E 00009             MOVAB   DBG$NPARSE_EXPRESSION, R10                   ;
                59 00000000G  00   9E 00010             MOVAB   DBG$NMATCH, R9                               ;
                58 00000000'  EF   9E 00017             MOVAB   DBG$CS_CR, R8                                ;
                              04   DD 0001E             PUSHL   #4                                           ; 0943
                6B            01   FB 00020             CALLS   #1, DBG$GET_TEMPMEM                          ;
                52            50   D0 00023             MOVL    R0, NOUN_NODE                                ;
                50       08   AC   D0 00026             MOVL    VERB_NODE, R0                                ; 0944
          08    A0            52   D0 0002A             MOVL    NOUN_NODE, 8(R0)                             ;
                54       0C   AC   D0 0002E             MOVL    MESSAGE_VECT, R4                             ; 0953
                              14   BB 00032             PUSHR   #^M<R2,R4>                                   ; 0952
                53       04   AC   D0 00034             MOVL    INPUT_DESC, R3                               ; 0951
                              53   DD 00038             PUSHL   R3                                           ; 0952
   00000000G    00            03   FB 0003A             CALLS   #3, DBG$NREAD_NAME                           ;
                03            50   E8 00041             BLBS    R0, 2$                                       ;
                            00FD   31 00044 1$:         BRW     13$                                          ;
                              01   DD 00047 2$:         PUSHL   #1                                           ; 0959
                         02   A8   9F 00049             PUSHAB  DBG$CS_EQUAL                                 ;
                              53   DD 0004C             PUSHL   R3                                           ;
                69            03   FB 0004E             CALLS   #3, DBG$NMATCH                               ;
                3D            50   E9 00051             BLBC    R0, 3$                                       ;
                57       08   A2   9E 00054             MOVAB   8(R2), LINK                                  ; 0965
```

```
                                04  DD 00058        PUSHL   #4                                  ; 0966
       6B                       01  FB 0005A        CALLS   #1, DBG$GET_TEMPMEM                 ;
       52                       50  D0 0005D        MOVL    R0, NOUN_NODE                       ;
       67                       52  D0 00060        MOVL    NOUN_NODE, (LINK)                   ; 0967
       56 00000000G             00  9A 00063        MOVZBL  DBG$GB_RADIX, RADIX                 ; 0971
                                54  DD 0006A        PUSHL   R4                                  ; 0978
                                0D  DD 0006C        PUSHL   #13                                 ; 0977
                                52  DD 0006E        PUSHL   NOUN_NODE                           ;
                    0048        8F  BB 00070        PUSHR   #^M<R3,R6>                          ;
       6A                       05  FB 00074        CALLS   #5, DBG$NPARSE_EXPRESSION           ;
       55                       50  D0 00077        MOVL    R0, STATUS                          ;
       01                       55  D1 0007A        CMPL    STATUS, #1                          ; 0986
                                1E  13 0007D        BEQL    4$                                  ;
       04                       55  D1 0007F        CMPL    STATUS, #4                          ; 0995
                                C0  13 00082        BEQL    1$                                  ;
                                02  DD 00084        PUSHL   #2                                  ; 1001
                      0C        A8  9F 00086        PUSHAB  DBG$CS_TO                           ;
                                53  DD 00089        PUSHL   R3                                  ;
       69                       03  FB 0008B        CALLS   #3, DBG$NMATCH                      ;
       12                       50  E8 0008E        BLBS    R0, 6$                              ;
                                01  DD 00091 3$:    PUSHL   #1                                  ; 1002
                    0108        8F  BB 00093        PUSHR   #^M<R3,R8>                          ;
       69                       03  FB 00097        CALLS   #3, DBG$NMATCH                      ;
       03                       50  E9 0009A        BLBC    R0, 5$                              ;
                              0080  31 0009D 4$:    BRW     10$                                 ;
                              008C  31 000A0 5$:    BRW     11$                                 ;
                                54  DD 000A3 6$:    PUSHL   R4                                  ; 1010
                                0E  DD 000A5        PUSHL   #14                                 ; 1009
                      0C        A2  9F 000A7        PUSHAB  12(NOUN_NODE)                       ;
                    0048        8F  BB 000AA        PUSHR   #^M<R3,R6>                          ;
       6A                       05  FB 000AE        CALLS   #5, DBG$NPARSE_EXPRESSION           ;
       55                       50  D0 000B1        MOVL    R0, STATUS                          ;
       01                       55  D1 000B4        CMPL    STATUS, #1                          ; 1019
                                67  13 000B7        BEQL    10$                                 ;
       04                       55  D1 000B9        CMPL    STATUS, #4                          ; 1028
                                86  13 000BC        BEQL    1$                                  ;
                                02  DD 000BE        PUSHL   #2                                  ; 1034
                      06        A8  9F 000C0        PUSHAB  DBG$CS_BY                           ;
                                53  DD 000C3        PUSHL   R3                                  ;
       69                       03  FB 000C5        CALLS   #3, DBG$NMATCH                      ;
       1D                       50  E9 000C8        BLBC    R0, 7$                              ;
                                54  DD 000CB        PUSHL   R4                                  ; 1042
                                05  DD 000CD        PUSHL   #5                                  ; 1041
                      04        A2  9F 000CF        PUSHAB  4(NOUN_NODE)                        ;
                    0048        8F  BB 000D2        PUSHR   #^M<R3,R6>                          ;
       6A                       05  FB 000D6        CALLS   #5, DBG$NPARSE_EXPRESSION           ;
       55                       50  D0 000D9        MOVL    R0, STATUS                          ;
       01                       55  D1 000DC        CMPL    STATUS, #1                          ; 1050
                                3F  13 000DF        BEQL    10$                                 ;
       04                       55  D1 000E1        CMPL    STATUS, #4                          ; 1059
                                05  12 000E4        BNEQ    8$                                  ;
                                5C  11 000E6        BRB     13$                                 ; 1061
                      04        A2  D4 000E8 7$:    CLRL    4(NOUN_NODE)                        ; 1066
                                02  DD 000EB 8$:    PUSHL   #2                                  ; 1070
                      09        A8  9F 000ED        PUSHAB  DBG$CS_DO                           ;
                                53  DD 000F0        PUSHL   R3                                  ;
       69                       03  FB 000F2        CALLS   #3, DBG$NMATCH                      ;
```

```
                              1C            50  E9 000F5        BLBC    R0, 9$                                       :
                              57      08    A2  9E 000F8        MOVAB   8(R2), LINK                                  : 1076
                                            04  DD 000FC        PUSHL   #4                                           : 1077
                              68            01  FB 000FE        CALLS   #1, DBG$GET_TEMPMEM                          :
                              52            50  D0 00101        MOVL    R0, NOUN_NODE                                :
                              67            52  D0 00104        MOVL    NOUN_NODE, (LINK)                            : 1078
                                            01  DD 00107        PUSHL   #1                                           : 1082
                                      04    A8  9F 00109        PUSHAB  DBG$CS_LEFT_PAREN                            :
                                            53  DD 0010C        PUSHL   R3                                           :
                              69            03  FB 0010E        CALLS   #3, DBG$NMATCH                               :
                              34            50  E8 00111        BLBS    R0, 14$                                      :
                                            01  DD 00114 9$:    PUSHL   #1                                           : 1083
                                    0108    8F  BB 00116        PUSHR   #^M<R3,R8>                                   :
                              69            03  FB 0011A        CALLS   #3, DBG$NMATCH                               :
                              0F            50  E9 0011D        BLBC    R0, 11$                                      :
                                000280D0    8F  DD 00120 10$:   PUSHL   #164048                                      :
                   00000000G  00            01  FB 00126        CALLS   #1, DBG$NMAKE_ARG_VECT                       :
                                            12  11 0012D        BRB     12$                                          :
                                            53  DD 0012F 11$:   PUSHL   R3                                           :
                   00000000G  00            01  FB 00131        CALLS   #1, DBG$NNEXT_WORD                           :
                                            50  DD 00138        PUSHL   R0                                           :
                   00000000G  00            01  FB 0013A        CALLS   #1, DBG$NSYNTAX_ERROR                        :
                              64            50  D0 00141 12$:   MOVL    R0, (R4)                                     :
                              50            04  D0 00144 13$:   MOVL    #4, R0                                       :
                                                04 00147        RET                                                  :
                                            52  DD 00148 14$:   PUSHL   NOUN_NODE                                    : 1094
                                            53  DD 0014A        PUSHL   R3                                           :
                   00000000G  00            02  FB 0014C        CALLS   #2, DBG$NSAVE_BREAK_BUFFER                   :
                              50            01  D0 00153        MOVL    #1, R0                                       : 1098
                                                04 00156        RET                                                  : 1100

; Routine Size:  343 bytes,    Routine Base:  DBG$CODE + 0315
```

```
1  GLOBAL ROUTINE dbg$nexecute_for (verb_node,message_vect) =
1  !++
1  ! Functional Description
1  !
1  !       This routine performs the action associated with the FOR
1  !       command.
1  !
1  ! Formal Parameters
1  !
1  !       verb_node       - A longword containing the address of the
1  !                         head (verb) node.
1  !       message_vect    - The address of a longword to contain the
1  !                         address of an error message vector
1  !
1  ! Implicit Inputs
1  !
1  !       The command tree contains a verb node and a linked list
1  !       of three noun nodes. (See the diagram in the header for
1  !       DBG$NPARSE_FOR).
1  !
1  ! Routine Value
1  !
1  !       A completion code.
1  !
1  ! Completion Codes
1  !
1  !       sts$k_success (1)       - Success. Command executed
1  !       sts$k_severe (4)        - Failure. The command could not be
1  !                                 executed. An error message is constructed.
1  !
1  ! Side Effects
1  !
1  !       None
1  !--
2      BEGIN
2
2      MAP
2          verb_node : REF dbg$verb_node;
2
2      LOCAL
2          action_node:    REF dbg$noun_node,       ! The noun node for the action clause
2          action_string:  REF VECTOR[,WORD],       ! Counted string with the action clause
2          bounds_node: REF dbg$noun_node,          ! Noun node with the upper and
2                                                   !     lower bounds
2          dummy,                                   ! Dummy output parameter
2          loop_incr,                               ! Loop increment
2          lower_bound,                             ! An integer with the lower
2                                                   !     loop bound
2          new_valdesc: REF dbg$valdesc,            ! A copy of a value descriptor
2          new_varname,                             ! Pointer to a copy of the
2                                                   !     variable name
2          symid_list,                              ! Points to a list of symids
2          upper_bound,                             ! An integer with the upper
2                                                   !     loop bound
2          valdesc: REF dbg$valdesc,                ! A pointer to a value
2                                                   !     descriptor
2          var_node: REF dbg$noun_node,             ! The noun node with the loop
```

```
                                                        !     variable
    var_name: REF VECTOR[,BYTE],                        ! The counted string with the
                                                        !     variable name
    vax_desc:        dbg$stg_desc;                      ! Target of the conversion from
                                                        !     the value descriptor
                                                        !     representing the count.
! Recover the noun nodes.
!
var_node = .verb_node [dbg$l_verb_object_ptr];
var_name = .var_node [dbg$l_noun_value];
bounds_node = .var_node [dbg$l_noun_link];
valdesc = .bounds_node [dbg$l_noun_value];
action_node = .bounds_node [dbg$l_noun_link];
action_string = .action_node [dbg$l_noun_value];

! Set up the vax descriptor for the bounds.
! This vax descriptor is of type integer longword, and is used to convert the
! language specific value descriptor for loop bounds to an
! integer quantity that we can use in a language-independent way.
!
vax_desc [dsc$b_class] = dsc$k_class_s;
vax_desc [dsc$b_dtype] = dsc$k_dtype_l;
vax_desc [dsc$w_length] = 4;
vax_desc [dsc$a_pointer] = lower_bound;

! Do the conversion from value descriptor to integer.
!
IF NOT dbg$ntype_conv (.valdesc,
                       dbg$k_default,
                       dbg$k_vax_desc,
                       vax_desc,
                       .message_vect)
THEN
    RETURN sts$k_severe;

! Do the conversion again, this time picking up the upper bound.
!
vax_desc [dsc$a_pointer] = upper_bound;
IF NOT dbg$ntype_conv (.bounds_node [dbg$l_noun_value2],
                       dbg$k_default,
                       dbg$k_vax_desc,
                       vax_desc,
                       .message_vect)
THEN
    RETURN sts$k_severe;

! Do the conversion once again, this time with the loop increment.
!
IF .bounds_node [dbg$l_adjective_ptr] EQL 0
THEN
    loop_incr = 1
ELSE
    BEGIN
    vax_desc [dsc$a_pointer] = loop_incr;
    IF NOT dbg$ntype_conv (.bounds_node [dbg$l_adjective_ptr],
                           dbg$k_default,
                           dbg$k_vax_desc,
                                vax_desc,
                                .message_vect)
            THEN
                RETURN sts$k_severe;
            END;

        ! If the loop increment is zero then signal an error.
        !
        IF .loop_incr EQL 0
        THEN
            SIGNAL (dbg$_loopincr);

        ! If the upper bound is below the lower bound, do nothing.
        !
        IF (.loop_incr GTR 0 AND .upper_bound LSS .lower_bound)
        OR (.loop_incr LSS 0 AND .upper_bound GTR .lower_bound)
        THEN
            RETURN sts$k_success;

        ! Make a value descriptor for the initial value of the loop variable.
        !
        new_valdesc = dbg$get_memory (dbg$k_valdesc_base_size+4);
        new_valdesc[dbg$w_dhdr_length] = (dbg$k_valdesc_base_size * 4) + 16;
        new_valdesc[dbg$b_dhdr_type] = dbg$k_value_desc;
        new_valdesc[dbg$b_dhdr_lang] = .dbg$gb_language;
        new_valdesc[dbg$b_dhdr_kind] = rst$k_data;
        new_valdesc[dbg$b_dhdr_fcode] = rst$k_type_atomic;
        new_valdesc[dbg$b_value_class] = dsc$k_class_s;
        new_valdesc[dbg$b_value_dtype] = dsc$k_dtype_l;
        new_valdesc[dbg$w_value_length] = 4;
        new_valdesc[dbg$l_value_pointer] = new_valdesc[dbg$l_value_value0];
        new_valdesc[dbg$l_value_value0] = .lower_bound;

        ! Also make a copy of the variable name. This is because the original
        ! varname pointer is being saved away by dbg$ncis_add and we don't
        ! want to free it twice.
        !
        new_varname = dbg$get_memory (1+.var_name[0]/4);
        ch$move (1+.var_name[0], .var_name, .new_varname);
        IF NOT dbg$def_sym_add (.new_varname, define_value,
                               .new_valdesc,
                               FALSE, dummy, .message_vect)
        THEN
            RETURN sts$k_severe;

        ! Add a link to the command input stream, containing the action
        ! string and the upper bound.
        !
        IF NOT dbg$ncis_add (action_string[1], .action_string[0], cis_for,
                    .upper_bound, .var_name, .loop_incr, .message_vect)
        THEN
            RETURN sts$k_severe;

        ! Return success.
        !
        RETURN sts$k_success;
```

```
; 1146         1272  1        END; ! dbg$nexecute_repeat
; INFO#250                  L1:1229
; Referenced LOCAL symbol UPPER_BOUND is probably not initialized
; INFO#250                  L1:1229
; Referenced LOCAL symbol LOWER_BOUND is probably not initialized



                                        0FFC 00000          .ENTRY  DBG$NEXECUTE_FOR, Save R2,R3,R4,R5,R6,R7,-   ; 1101
                                                                    R8,R9,R10,R11                                 ;
                     5E           1C   C2 00002             SUBL2   #28, SP                                       ;
                     50      04   AC   D0 00005             MOVL    VERB_NODE, R0                                 ; 1166
                     50      08   A0   D0 00009             MOVL    8(R0), VAR_NODE                               ;
                     56           60   D0 0000D             MOVL    (VAR_NODE), VAR_NAME                          ; 1167
                     52      08   A0   D0 00010             MOVL    8(VAR_NODE), BOUNDS_NODE                      ; 1168
                     51           62   D0 00014             MOVL    (BOUNDS_NODE), VALDESC                        ; 1169
                     50      08   A2   D0 00017             MOVL    8(BOUNDS_NODE), ACTION_NODE                   ; 1170
                     5B           60   D0 0001B             MOVL    (ACTION_NODE), ACTION_STRING                  ; 1171
          10         AE 01080004  8F   D0 0001E             MOVL    #17301508, VAX_DESC                           ; 1180
          14         AE           6E   9E 00026             MOVAB   LOWER_BOUND, VAX_DESC+4                       ; 1181
                     58      08   AC   D0 0002A             MOVL    MESSAGE_VECT, R8                              ; 1189
                                  58   DD 0002E             PUSHL   R8                                            ;
                             14   AE   9F 00030             PUSHAB  VAX_DESC                                      ; 1185
                     7E      82   8F   9A 00033             MOVZBL  #130, -(SP)                                   ;
                                  01   DD 00037             PUSHL   #1                                            ;
                                  51   DD 00039             PUSHL   VALDESC                                       ;
           00000000G 00           05   FB 0003B             CALLS   #5, DBG$NTYPE_CONV                            ;
                     42           50   E9 00042             BLBC    R0, 2$                                        ;
          14         AE      04   AE   9E 00045             MOVAB   UPPER_BOUND, VAX_DESC+4                       ; 1195
                                  58   DD 0004A             PUSHL   R8                                            ; 1200
                             14   AE   9F 0004C             PUSHAB  VAX_DESC                                      ; 1196
                     7E      82   8F   9A 0004F             MOVZBL  #130, -(SP)                                   ;
                                  01   DD 00053             PUSHL   #1                                            ;
                             0C   A2   DD 00055             PUSHL   12(BOUNDS_NODE)                               ;
           00000000G 00           05   FB 00058             CALLS   #5, DBG$NTYPE_CONV                            ;
                     25           50   E9 0005F             BLBC    R0, 2$                                        ;
                             04   A2   D5 00062             TSTL    4(BOUNDS_NODE)                                ; 1206
                                  06   12 00065             BNEQ    1$                                            ;
          08         AE           01   D0 00067             MOVL    #1, LOOP_INCR                                 ; 1208
                                  20   11 0006B             BRB     3$                                            ;
          14         AE      08   AE   9E 0006D  1$:        MOVAB   LOOP_INCR, VAX_DESC+4                         ; 1211
                                  58   DD 00072             PUSHL   R8                                            ; 1216
                             14   AE   9F 00074             PUSHAB  VAX_DESC                                      ; 1212
                     7E      82   8F   9A 00077             MOVZBL  #130, -(SP)                                   ;
                                  01   DD 0007B             PUSHL   #1                                            ;
                             04   A2   DD 0007D             PUSHL   4(BOUNDS_NODE)                                ;
           00000000G 00           05   FB 00080             CALLS   #5, DBG$NTYPE_CONV                            ;
                     03           50   E8 00087  2$:        BLBS    R0, 3$                                        ;
                                00AB   31 0008A             BRW     8$                                            ;
                     59      08   AE   D0 0008D  3$:        MOVL    LOOP_INCR, R9                                 ; 1223
                                  0D   12 00091             BNEQ    4$                                            ;
                        00028F18  8F   DD 00093             PUSHL   #167704                                       ; 1225
           00000000G 00           01   FB 00099             CALLS   #1, LIB$SIGNAL                                ;
                                  59   D5 000A0  4$:        TSTL    R9                                            ; 1229
                                  09   15 000A2             BLEQ    6$                                            ;
```

```
                           6E       04  AE  D1 000A4           CMPL    UPPER_BOUND, LOWER_BOUND            :
                                        03  18 000A8           BGEQ    6$                                  :
                                      008F  31 000AA 5$:       BRW     9$                                  :
                                        59  D5 000AD 6$:       TSTL    R9                                  : 1230
                                        06  18 000AF           BGEQ    7$                                  :
                           6E       04  AE  D1 000B1           CMPL    UPPER_BOUND, LOWER_BOUND            :
                                        F3  14 000B5           BGTR    5$                                  :
                                        0C  DD 000B7 7$:       PUSHL   #12                                 : 1236
                00000000G  00           01  FB 000B9           CALLS   #1, DBG$GET_MEMORY                  :
                           57           50  D0 000C0           MOVL    R0, NEW_VALDESC                     :
                           67           30  B0 000C3           MOVW    #48, (NEW_VALDESC)                  : 1237
                     02    A7       7A  8F  90 000C6           MOVB    #122, 2(NEW_VALDESC)                : 1238
                     03    A7 00000000G 00  90 000CB           MOVB    DBG$GB_LANGUAGE, 3(NEW_VALDESC)     : 1239
                     06    A7     0602  8F  B0 000D3           MOVW    #1538, 6(NEW_VALDESC)               : 1241
                     14    A7 01080004  8F  D0 000D9           MOVL    #17301508, 20(NEW_VALDESC)          : 1244
                     18    A7       20  A7  9E 000E1           MOVAB   32(NEW_VALDESC), 24(NEW_VALDESC)    : 1245
                     20    A7           6E  D0 000E6           MOVL    LOWER_BOUND, 32(NEW_VALDESC)        : 1246
                           50           66  9A 000EA           MOVZBL  (VAR_NAME), R0                      : 1252
                           50           04  C6 000ED           DIVL2   #4, R0                              :
                                    01  A0  9F 000F0           PUSHAB  1(R0)                               :
                00000000G  00           01  FB 000F3           CALLS   #1, DBG$GET_MEMORY                  :
                           5A           50  D0 000FA           MOVL    R0, NEW_VARNAME                     :
                           50           66  9A 000FD           MOVZBL  (VAR_NAME), R0                      : 1253
                                        50  D6 00100           INCL    R0                                  :
             6A            66           50  28 00102           MOVC3   R0, (VAR_NAME), (NEW_VARNAME)       :
                                        58  DD 00106           PUSHL   R8                                  : 1256
                                    10  AE  9F 00108           PUSHAB  DUMMY                               : 1254
                                        7E  D4 0010B           CLRL    -(SP)                               :
                                        57  DD 0010D           PUSHL   NEW_VALDESC                         : 1255
                                        05  DD 0010F           PUSHL   #5                                  : 1254
                                        5A  DD 00111           PUSHL   NEW_VARNAME                         :
                00000000G  00           06  FB 00113           CALLS   #6, DBG$DEF_SYM_ADD                 :
                           1B           50  E9 0011A           BLBC    R0, 8$                              :
                                        58  DD 0011D           PUSHL   R8                                  : 1264
                                  0240  8F  BB 0011F           PUSHR   #^M<R6,R9>                          :
                                    10  AE  DD 00123           PUSHL   UPPER_BOUND                         :
                                        07  DD 00126           PUSHL   #7                                  : 1263
                           7E           6B  3C 00128           MOVZWL  (ACTION_STRING), -(SP)              :
                                    02  AB  9F 0012B           PUSHAB  2(ACTION_STRING)                    :
                00000000G  00           07  FB 0012E           CALLS   #7, DBG$NCIS_ADD                    :
                           04           50  E8 00135           BLBS    R0, 9$                              :
                           50           04  D0 00138 8$:       MOVL    #4, R0                              : 1266
                                            04 0013B           RET                                         :
                           50           01  D0 0013C 9$:       MOVL    #1, R0                              : 1270
                                            04 0013F           RET                                         : 1272

; Routine Size:  320 bytes,    Routine Base:  DBG$CODE + 046C
```

```
1  GLOBAL ROUTINE dbg$nparse_repeat(input_desc, verb_node, message_vect) =
1  !
1  ! Functional Description
1  !
1  !       ATN parse network for the REPEAT verb.
1  !       This routine takes a verb node for the REPEAT verb, and a string
1  !       descriptor for the remaining (unparsed) input.
1  !       A command execution tree is built. The form of the tree is:
1  !
1  !        -----------        -----------        -----------
1  !       | verb node |-->--| noun node |-->--| noun node |
1  !        -----------        -----------        -----------
1  !
1  !       The first noun node points to a value descriptor for the count.
1  !       The second noun node points to a counted string with the action clause.
1  !
1  ! Formal Parameters
1  !
1  !       input_desc      - A longword containing the address of the
1  !                               command input descriptor.
1  !       verb_node       - A longword containing the address of the verb node.
1  !       message_vect    - The address of a longword to contain the address
1  !                               of a standard message argument vector.
1  !
1  ! Implicit Inputs
1  !
1  !       none
1  !
1  ! Implicit Outputs
1  !
1  !       On success, the command execution tree is constructed.
1  !       On failure, a message argument vector is constructed or obtained.
1  !
1  ! Routine value
1  !
1  !       sts$k_success (1)       - Success. Command execution tree constructed.
1  !       sts$k_severe  (4)       - Failure. Error encountered. Message argument
1  !                                 constructed and returned.
1  !
1  ! Side Effects
1  !
1  !       Permanent storage is allocated for the string holding the action
1  !       clause; this is released in DBG$NEXECUTE_REPEAT after execution
1  !       of the action clause.
1  !
1
2      BEGIN
2
2      MAP
2          input_desc  : REF dbg$stg_desc,
2          verb_node       : REF dbg$verb_node;
2
2      BIND
2          dbg$cs_cr               = UPLIT BYTE (1, dbg$k_car_return),
2          dbg$cs_left_paren       = UPLIT BYTE (1, dbg$k_left_parenthesis),
2          dbg$cs_do               = UPLIT BYTE (2, 'DO');
2
```

```
2      LOCAL
2          link,                              ! Temporary to hold links in the command
2                                             !     execution tree.
2          noun_node : REF dbg$noun_node,     ! A node in the command execution tree.
2          radix,                             ! Holds the current radix setting.
2          status;                            ! Holds return status from subroutine
2                                             !     calls.
2
2
2      ! Create and link a noun node
2      !
2      noun_node = dbg$get_tempmem(dbg$k_noun_node_size);
2      verb_node[dbg$l_verb_object_ptr] = .noun_node;
2
2      ! Determine the current radix.
2      !
2      radix  = .dbg$gb_radix[dbg$b_radix_input];
2
2      ! Obtain a value descriptor for the count. The first noun node
2      ! points to this descriptor.
2      !
2      STATUS = DBG$NPARSE_EXPRESSION(.INPUT_DESC, .RADIX,
2                                     NOUN_NODE [DBG$L_NOUN_VALUE],
2                                     TOKEN$K_TERM_DO, .MESSAGE_VECT);
2
2
2      ! The return status should be "warning", meaning that an expression
2      ! was parsed and further input reamins. If an expression was parsed
2      ! but no input remains, then DBG$NPARSE_EXPRESSION will return success.
2      ! In this context, it is an error since "REPEAT count" by itself
2      ! is an error.
2      !
2      IF .status EQL sts$k_success
2      THEN
3          BEGIN
3          .message_vect = dbg$nmake_arg_vect (dbg$_needmore);
3          RETURN sts$k_severe;
3          END;
2
2      ! Severe status is also an error.
2      !
2      IF .status EQL sts$k_severe
2      THEN
2          RETURN sts$k_severe;
2
2      ! Eat the DO.
2      !
2      IF NOT dbg$nmatch (.input_desc, dbg$cs_do, 1)
2      THEN
3          BEGIN
3          .message_vect =
4              (IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
4               THEN
4                  dbg$nmake_arg_vect (dbg$_needmore)
4               ELSE
3                  dbg$nsyntax_error (dbg$nnext_word (.input_desc)));
3          RETURN sts$k_severe;
```

```
2              END;
2
2       ! Allocate and link a noun node for the action clause.
2       !
2       link = noun_node [dbg$l_noun_link];
2       noun_node = dbg$get_tempmem(dbg$k_noun_node_size);
2       .link = .noun_node;
2
2       ! Eat the left parenthesis which we require be present.
2       !
2       IF NOT dbg$nmatch (.input_desc, dbg$cs_left_paren, 1)
2       THEN
3          BEGIN
3          .message_vect =
4              (IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
4               THEN
4                   dbg$nmake_arg_vect (dbg$_needmore)
4               ELSE
5                   BEGIN
5                   SIGNAL(dbg$_needparen);
5                   dbg$nsyntax_error (dbg$nnext_word (.input_desc))
5                   END);
3          RETURN sts$k_severe;
3          END;
2
2       ! Put a pointer to the counted string representing the action
2       ! clause into the second noun node. (Note - the counted string
2       ! is constructed out of "permanent" memory which is released
2       ! in DBG$NEXECUTE_REPEAT).
2       ! The third argument indicates that save_break_buffer is not being
2       ! called during parsing of a SET BREAK DO (The routine behaves
2       ! slightly differently in that case)
2       !
2       dbg$nsave_break_buffer (.input_desc, noun_node [dbg$l_noun_value]);
2
2       ! Return success.
2       !
2       RETURN sts$k_success;
2
1       END;



                                                              .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                    0D  01  00026 P.AAO:      .BYTE   1, 13                                  ;
                                    28  01  00028 P.AAP:      .BYTE   1, 40                                  ;
                                        02  0002A P.AAQ:      .BYTE   2                                      ;
                                    4F  44  0002B             .ASCII  \DO\                                   ;

                                                  DBG$CS_CR=          P.AAO
                                                  DBG$CS_LEFT_PAREN=  P.AAP
                                                  DBG$CS_DO=          P.AAQ


                                                              .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0
```

```
                                         00FC 00000          .ENTRY   DBG$NPARSE_REPEAT, Save R2,R3,R4,R5,R6,R7      ; 1273
                57 00000000G       00     9E 00002           MOVAB    DBG$GET_TEMPMEM, R7                            ;
                56 00000000G       00     9E 00009           MOVAB    DBG$NMATCH, R6                                 ;
                55 00000000'       EF     9E 00010           MOVAB    DBG$CS_CR, R5                                  ;
                                   04     DD 00017           PUSHL    #4                                             ; 1341
                67                 01     FB 00019           CALLS    #1, DBG$GET_TEMPMEM                            ;
                54                 50     D0 0001C           MOVL     R0, NOUN_NODE                                  ;
                50          08     AC     D0 0001F           MOVL     VERB_NODE, R0                                  ; 1342
      08        A0                 54     D0 00023           MOVL     NOUN_NODE, 8(R0)                               ;
                50 00000000G       00     9A 00027           MOVZBL   DBG$GB_RADIX, RADIX                            ; 1346
                            0C     AC     DD 0002E           PUSHL    MESSAGE_VECT                                   ; 1353
                                   05     DD 00031           PUSHL    #5                                             ; 1352
                                   11     BB 00033           PUSHR    #^M<R0,R4>                                     ;
                52          04     AC     D0 00035           MOVL     INPUT_DESC, R2                                 ; 1351
                                   52     DD 00039           PUSHL    R2                                             ; 1352
 00000000G      00                 05     FB 0003B           CALLS    #5, DBG$NPARSE_EXPRESSION                      ;
                53                 50     D0 00042           MOVL     R0, STATUS                                     ;
                01                 53     D1 00045           CMPL     STATUS, #1                                     ; 1362
                                   44     13 00048           BEQL     2$                                             ;
                04                 53     D1 0004A           CMPL     STATUS, #4                                     ; 1371
                                   71     13 0004D           BEQL     6$                                             ;
                                   01     DD 0004F           PUSHL    #1                                             ; 1377
                            04     A5     9F 00051           PUSHAB   DBG$CS_DO                                      ;
                                   52     DD 00054           PUSHL    R2                                             ;
                66                 03     FB 00056           CALLS    #3, DBG$NMATCH                                 ;
                0C                 50     E8 00059           BLBS     R0, 1$                                         ;
                                   01     DD 0005C           PUSHL    #1                                             ; 1381
                                   24     BB 0005E           PUSHR    #^M<R2,R5>                                     ;
                66                 03     FB 00060           CALLS    #3, DBG$NMATCH                                 ;
                44                 50     E9 00063           BLBC     R0, 4$                                         ;
                                   26     11 00066           BRB      2$                                             ; 1383
                53          08     A4     9E 00068  1$:      MOVAB    8(R4), LINK                                    ; 1391
                                   04     DD 0006C           PUSHL    #4                                             ; 1392
                67                 01     FB 0006E           CALLS    #1, DBG$GET_TEMPMEM                            ;
                54                 50     D0 00071           MOVL     R0, NOUN_NODE                                  ;
                63                 54     D0 00074           MOVL     NOUN_NODE, (LINK)                              ; 1393
                                   01     DD 00077           PUSHL    #1                                             ; 1397
                            02     A5     9F 00079           PUSHAB   DBG$CS_LEFT_PAREN                              ;
                                   52     DD 0007C           PUSHL    R2                                             ;
                66                 03     FB 0007E           CALLS    #3, DBG$NMATCH                                 ;
                40                 50     E8 00081           BLBS     R0, 7$                                         ;
                                   01     DD 00084           PUSHL    #1                                             ; 1401
                                   24     BB 00086           PUSHR    #^M<R2,R5>                                     ;
                66                 03     FB 00088           CALLS    #3, DBG$NMATCH                                 ;
                0F                 50     E9 0008B           BLBC     R0, 3$                                         ;
                      000280D0     8F     DD 0008E  2$:      PUSHL    #164048                                        ; 1403
 00000000G      00                 01     FB 00094           CALLS    #1, DBG$NMAKE_ARG_VECT                         ;
                                   1F     11 0009B           BRB      5$                                             ;
                      00028743     8F     DD 0009D  3$:      PUSHL    #165699                                        ; 1406
 00000000G      00                 01     FB 000A3           CALLS    #1, LIB$SIGNAL                                 ;
                                   52     DD 000AA  4$:      PUSHL    R2                                             ; 1407
 00000000G      00                 01     FB 000AC           CALLS    #1, DBG$NNEXT_WORD                             ;
                                   50     DD 000B3           PUSHL    R0                                             ;
 00000000G      00                 01     FB 000B5           CALLS    #1, DBG$NSYNTAX_ERROR                          ;
      0C        BC                 50     D0 000BC  5$:      MOVL     R0, @MESSAGE_VECT                              ; 1401
                50                 04     D0 000C0  6$:      MOVL     #4, R0                                         ; 1409
                                          04 000C3           RET                                                     ;
```

```
                                              14  BB 000C4 7$:     PUSHR   #^M<R2,R4>                                  ; 1420
                         00000000G  00        02  FB 000C6         CALLS   #2, DBG$NSAVE_BREAK_BUFFER                 ;
                                    50        01  D0 000CD         MOVL    #1, R0                                      ; 1424
                                                  04 000D0         RET                                                 ; 1426

; Routine Size:  209 bytes,    Routine Base:  DBG$CODE + 05AC
```

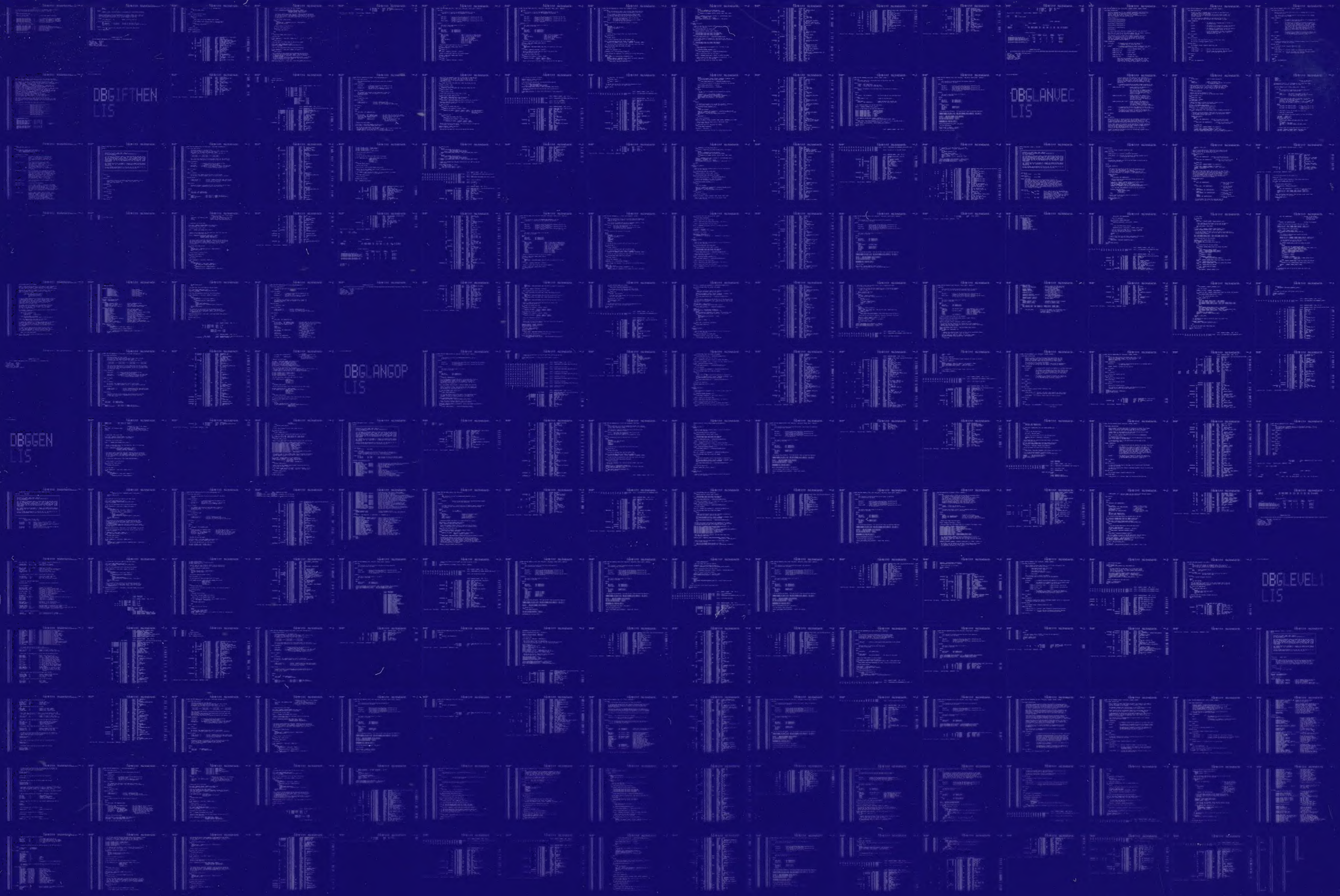
```
1 GLOBAL ROUTINE dbg$nexecute_repeat (verb_node,message_vect) =
1 !++
1 ! Functional Description
1 !
1 !       This routine performs the action associated with the REPEAT
1 !       command.
1 !
1 ! Formal Parameters
1 !
1 !       verb_node       - A longword containing the address of the
1 !                         head (verb) node.
1 !       message_vect    - The address of a longword to contain the
1 !                         address of an error message vector
1 !
1 ! Implicit Inputs
1 !
1 !       The command tree contains a verb node and a linked list
1 !       of two noun nodes. (See the diagram in the header for
1 !       DBG$NPARSE_REPEAT).
1 !
1 ! Routine Value
1 !
1 !       A completion code.
1 !
1 ! Completion Codes
1 !
1 !       sts$k_success (1)       - Success. Command executed
1 !       sts$k_severe (4)        - Failure. The command could not be
1 !                                 executed. An error message is constructed.
1 !
1 ! Side Effects
1 !
1 !       None
1 !--
2     BEGIN
2
2     MAP
2         verb_node : REF dbg$verb_node;
2
2     LOCAL
2         action_node:    REF dbg$noun_node,        ! The noun node for the action clause
2         action_string:  REF VECTOR[,WORD],        ! Counted string with the action clause
2         count_node: REF dbg$noun_node,            ! The noun node for the count
2         count_value,                              ! The actual count
2         vax_desc:       dbg$stg_desc;             ! Target of the conversion from
2                                                   !    the value descriptor
2                                                   !    representing the count.
2
2     ! Recover the noun nodes.
2     !
2     count_node = .verb_node [dbg$l_verb_object_ptr];
2     action_node = .count_node [dbg$l_noun_link];
2
2     ! Set up the vax descriptor for the count.
2     ! This vax descriptor is of type integer longword, and is used to convert the
2     ! language specific value descriptor for a count to an
2     ! integer quantity that we can use in a language-independent way.
```

```
; 1360          1484  2        !
; 1361          1485  2        vax_desc [dsc$b_class] = dsc$k_class_s;
; 1362          1486  2        vax_desc [dsc$b_dtype] = dsc$k_dtype_l;
; 1363          1487  2        vax_desc [dsc$w_length] = 4;
; 1364          1488  2        vax_desc [dsc$a_pointer] = count_value;
; 1365          1489  2
; 1366          1490  2        ! Initialize count_value to 0
; 1367          1491  2        !
; 1368          1492  2        count_value = 0;
; 1369          1493  2
; 1370          1494  2        ! Do the conversion from value descriptor to boolean.
; 1371          1495  2        !
; 1372          1496  2        IF NOT dbg$ntype_conv (.count_node [dbg$l_noun_value],
; 1373          1497  2                               dbg$k_default,
; 1374          1498  2                               dbg$k_vax_desc,
; 1375          1499  2                               vax_desc,
; 1376          1500  2                               .message_vect)
; 1377          1501  2        THEN
; 1378          1502  2            RETURN sts$k_severe;
; 1379          1503  2
; 1380          1504  2        ! Recover the string.
; 1381          1505  2        !
; 1382          1506  2        action_string = .action_node [dbg$l_noun_value];
; 1383          1507  2
; 1384          1508  2        ! Add a link to the command input stream, containing the action
; 1385          1509  2        ! string and the repeat count.
; 1386          1510  2        !
; 1387          1511  2        IF NOT dbg$ncis_add (action_string[1], .action_string[0], cis_repeat,
; 1388          1512  2                    .count_value, 0, 0, .message_vect)
; 1389          1513  2        THEN
; 1390          1514  2            RETURN sts$k_severe;
; 1391          1515  2
; 1392          1516  2        ! Return success.
; 1393          1517  2        !
; 1394          1518  2        RETURN sts$k_success;
; 1395          1519  2
; 1396          1520  1        END; ! dbg$nexecute_repeat



                                       0004 00000        .ENTRY  DBG$NEXECUTE_REPEAT, Save R2                  ; 1427
                 5E               10    C2 00002        SUBL2   #16, SP                                        ;
                 50        04     AC    D0 00005        MOVL    VERB_NODE, R0                                  ; 1477
                 50        08     A0    D0 00009        MOVL    8(R0), COUNT_NODE                              ;
                 52        08     A0    D0 0000D        MOVL    8(COUNT_NODE), ACTION_NODE                     ; 1478
        04       AE  01080004     8F    D0 00011        MOVL    #17301508, VAX_DESC                            ; 1487
        08       AE               6E    9E 00019        MOVAB   COUNT_VALUE, VAX_DESC+4                        ; 1488
                                  6E    D4 0001D        CLRL    COUNT_VALUE                                    ; 1492
                           08     AC    DD 0001F        PUSHL   MESSAGE_VECT                                   ; 1500
                           08     AE    9F 00022        PUSHAB  VAX_DESC                                       ; 1496
                 7E        82     8F    9A 00025        MOVZBL  #130, -(SP)                                    ;
                                  01    DD 00029        PUSHL   #1                                             ;
                                  60    DD 0002B        PUSHL   (COUNT_NODE)                                   ;
00000000G        00               05    FB 0002D        CALLS   #5, DBG$NTYPE_CONV                             ;
                 1D               50    E9 00034        BLBC    R0, 1$                                         ;
```

```
                        50              62  D0 00037          MOVL    (ACTION_NODE), ACTION_STRING                        ; 1506
                                08      AC  DD 0003A          PUSHL   MESSAGE_VECT                                        ; 1512
                                        7E  7C 0003D          CLRQ    -(SP)                                               ; 1511
                                0C      AE  DD 0003F          PUSHL   COUNT_VALUE                                         ; 1512
                                        04  DD 00042          PUSHL   #4                                                  ; 1511
                        7E              60  3C 00044          MOVZWL  (ACTION_STRING), -(SP)                              ;
                                02      A0  9F 00047          PUSHAB  2(ACTION_STRING)                                    ;
             00000000G  00              07  FB 0004A          CALLS   #7, DBG$NCIS_ADD                                    ;
                        04              50  E8 00051          BLBS    R0, 2$                                              ;
                        50              04  D0 00054 1$:      MOVL    #4, R0                                              ; 1514
                                            04 00057          RET                                                         ;
                        50              01  D0 00058 2$:      MOVL    #1, R0                                              ; 1518
                                            04 0005B          RET                                                         ; 1520

; Routine Size:  92 bytes,    Routine Base:  DBG$CODE + 067D


; 1397          1521  1 END
; 1398          1522  0 ELUDOM




                                                              .EXTRN  LIB$SIGNAL

;                                 PSECT SUMMARY
;
;       Name                       Bytes                          Attributes
;
;  DBG$PLIT                           45  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$CODE                         1753  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)



;                            Library Statistics
;
;                                           -------- Symbols --------      Pages        Processing
;       File                                Total    Loaded   Percent      Mapped       Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1             18619        9         0        1000         00:01.8
;  _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1        32        0         0           7         00:00.1
;  _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1        1545       97         6          97         00:01.9
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                              418        2         0          31         00:00.4
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1         386        3         0          22         00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGGEN.L32;1         150        0         0          12         00:00.3



; Information:   2
; Warnings:      0
; Errors:        0
;                                             COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:DBGIFTHEN/OBJ=OBJ$:DBGIFTHEN MSRC$:DBGIFTHEN/UPDATE=(ENH$:DBGIFTHEN)

; Size:          1753 code + 45 data bytes
; Run Time:         00:37.2
; Elapsed Time:     01:43.6
; Lines/CPU Min:      2454
; Lexemes/CPU-Min: 11235
; Memory Used:  186 pages
; Compilation Complete
```

0084 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY


DBGIFTHEN
LIS

DBGLANVEC
LIS

DBGLANGOP
LIS

DBGGEN
LIS

DBGLEVEL1
LIS